EFFICIENT LLL-BASED LATTICE REDUCTION FOR MIMO DETECTION: FROM ALGORITHMS TO IMPLEMENTATIONS

A Thesis
Presented to
The Academic Faculty

by

Qingsong Wen

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2017

EFFICIENT LLL-BASED LATTICE REDUCTION FOR MIMO DETECTION: FROM ALGORITHMS TO IMPLEMENTATIONS

Approved by:

Dr. Xiaoli Ma, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Geoffrey Ye Li
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Gee-Kung Chang
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Yao Xie
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Robert J. Baxley
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: March 27, 2017

To my dear family, my advisor, and my friends.

ACKNOWLEDGEMENTS

First, I would like to express my deepest gratitude to my advisor, Dr. Xiaoli Ma, for her persistent encouragement and support during my Ph.D. study and research at Georgia Institute of Technology. I benefit a lot from her unique training program that has inspired me a lot and helped me overcome challenges in the research.

Second, my sincere thanks go to the other members of my Ph.D. dissertation committee: Dr. Gee-Kung Chang, Dr. Robert J. Baxley, Dr. Geoffrey Ye Li, and Dr. Yao Xie, for serving on my committee and providing me valuable comments to improve my dissertation, which are of great help.

Third, I would like to thank the members in our research group: Dr. Sungeun Lee, Dr. Qi Zhou, Dr. Yiyin Wang, Dr. Wei Zhang, Dr. Giwan Choi, Dr. Benjamin R. Hamilton, Dr. Hayang Kim, Dr. Malik Muhammad Usman Gul, Dr. Marie Shinotsuka, Dr. Zhenhua Yu, Dr. Lingchen Zhu, Dr. Kai Ying, Dr. Andrew Harper, Dr. Brian Beck, Yiming Kong, Hyunwoo Cho and other colleagues, for the support and help during these years. I would also like to thank my dear friends at Georgia Institute of Technology, Dr. Lu lu, Dr. Cong Xiong, Dr. Chong Han, Dr. Jing Wang, Dr. Yun Wei, Dr. Yangfeng Ji, Yipu Zhao and my other friends.

In the end, I would like to thank my parents, my wife, and my children. My parents and my wife have been very helpful and supportive throughout my Ph.D. studies. Without their care and love, I could not have had such a happy and fruitful life. I would also like to thank my two lovely children, who are always the source of happiness in my family.

TABLE OF CONTENTS

DEDICATION . . . . . **iii**

ACKNOWLEDGEMENTS . . . . . **iv**

LIST OF TABLES . . . . . **viii**

LIST OF FIGURES . . . . . **ix**

SUMMARY . . . . . **xi**

I INTRODUCTION **1**

1.1 Motivations 1

1.2 Objectives 3

1.3 Outline 4

1.4 Notations 4

II BACKGROUND **6**

2.1 MIMO Detection 6

2.1.1 Conventional MIMO Detection 6

2.1.2 Lattice-Reduction-Aided MIMO Detectors 8

2.2 LLL Lattice Reduction Algorithms 10

2.2.1 LLL and ELLL Algorithms 10

2.2.2 Greedy and Fixed-Complexity LLL Algorithms 11

2.3 Existing Hardware Implementations of Lattice Reduction Algorithms 15

III PERFORMANCE ANALYSIS OF LR-AIDED MIMO DETECTORS **17**

3.1 Error Performance Analysis of DLLL-aided LDs 17

3.2 Error Performance Analysis of LLL-aided SIC/K-best Detectors . . 21

IV ENHANCED GREEDY LLL ALGORITHMS **22**

4.1 Finding the Candidate Set of LLL Iterations: Relaxing the Lovász Condition 22

4.2 Selecting the LLL Iteration: Relaxing the Decrescence of LLL Potential or Improving the Error Performance under Limited Iterations . . 23
4.3 Summary of the Two Enhanced Greedy LLL Algorithms 25
4.4 Numerical Results and Discussion 28
4.4.1 BER Comparisons of Different LR-aided MIMO Detectors . . 29
4.4.2 Convergence Comparisons of Different LR Algorithms 32
4.4.3 Complexity Comparisons of Different LR Algorithms 34
4.4.4 BER, Convergence, and Complexity of Different LR Algorithms with Early Termination 36

V INCREMENTAL FIXED-COMPLEXITY LLL ALGORITHMS 41

5.1 Incremental Column Traverse Strategies 41
5.2 Improved Termination Criterion 44
5.3 Numerical Results 47
5.3.1 Complexity Comparisons of Different LR Algorithms 47
5.3.2 BER Comparisons of Different LR-aided MIMO Detectors . . 49
5.3.3 BER Comparisons of Different LR-aided Detectors in Large MIMO Systems 53

VI HARDWARE-ORIENTED INCREMENTAL FCLLL ALGORITHM AND FIXED-POINT DESIGN 55

6.1 Hardware-Oriented Incremental fcLLL Algorithm 55
6.1.1 Simplified Size Reduction and Siegel Condition 56
6.1.2 Proposed Two-Angle CGR for Column Swap 57
6.2 Fixed-point Design with Wordlength Optimization 58

VII ITERATIVE HARDWARE IMPLEMENTATION OF INCREMENTAL FCLLL ALGORITHM 64

7.1 Proposed Iterative Hardware Architecture 64
7.1.1 Datapath Design 65
7.1.2 Control Path Design, Data Exchange, and Transfer 72
7.2 Implementation and Comparison 74
7.2.1 Performance Comparison in MIMO Detection 75

7.2.2 Performance Comparison of FPGA Implementations 76
7.3 Conclusion 78
VIII PIPELINING HARDWARE IMPLEMENTATION OF INCREMENTAL FCLLL ALGORITHM 79
8.1 Proposed Pipelined Hardware Architecture 79
8.1.1 Architecture Design of the Two-Angle CGR Module for Column Swap 80
8.1.2 Modules Cooperation for Processing Matrices 82
8.1.3 Timing Schedule and Data Transfer 85
8.2 Implementation and Comparison 91
8.2.1 Implementation Results 91
8.2.2 Comparison with FPGA Implementations 92
8.2.3 Comparison with ASIC and ASIP Implementations 93
8.3 Conclusion 96
IX CONCLUDING REMARKS 97
9.1 Contributions 97
9.2 Suggestions for Future Research 98
9.3 Publications 99
APPENDIX A — PROOF OF PROPOSITION 2.1 102
APPENDIX B — PROOF OF PROPOSITION 3.1 105
APPENDIX C — PROOF OF PROPOSITION 4.1 106
APPENDIX D — PROOF OF PROPOSITION 4.2 109
REFERENCES 111
VITA 120

LIST OF TABLES

2.1 The LLL/ELLL Algorithm 12

2.2 The Unified Form of Sequential FcLLL and Even-odd FcLLL 16

3.1 Equivalent Form of the LLL Algorithm and LLL Variants 19

4.1 Proposed Greedy LLL Algorithm-I 26

4.2 Proposed Greedy LLL Algorithm-II 27

4.3 The average and worst case numbers of LLL iterations in 4×4 and 8×8 MIMO systems. 34

5.1 The Incremental fcLLL Algorithm with Fixed Number of Iterations . 45

5.2 The Incremental fcLLL Algorithm with Maximum Number of Iterations 46

6.1 Hardware-oriented Incremental fcLLL Algorithm 56

7.1 Comparison of FPGA Implementations of the LLL Variants for 4×4 MIMO systems. 77

8.1 Distribution of FPGA Slices for Different Module 92

8.2 Comparison of FPGA Implementations of the LLL Variants for 4×4 MIMO Systems. 95

8.3 Comparison of ASIC/ASIP Implementations of the LLL Variants for 4×4 MIMO Systems. 95

LIST OF FIGURES

2.1 Convergence comparisons of different LR algorithms in 4×4 and 8×8 MIMO systems. For each LR algorithm, MMSE-SQRD is adopted, the parameter $\delta = 0.75$, and 10^6 i.i.d. Gaussian channel realizations are simulated. 14

3.1 The change of matrix $\tilde{\boldsymbol{R}}^{\star}$ with different LLL iteration selection when $N_t = 4$. *Two* column norms may be reduced if selecting the LLL iteration with $k = 3$, while *three* column norms may be reduced if selecting the LLL iteration with $k = 2$. 20

4.1 BER performance comparisons of dual-LR-aided MMSE and LR-aided MMSE-SIC detectors for 4×4 and 8×8 MIMO systems with 64-QAM. 31

4.2 Convergence comparisons of different LR algorithms by CCDF of LLL iterations in 4×4 and 8×8 MIMO systems. 33

4.3 Complexity comparisons of different LR algorithms from 3×3 to 8×8 MIMO systems. 35

4.4 BER comparisons of different LR algorithms with early termination in a 4×4 MIMO system with 64-QAM. 38

4.5 Complexity versus maximum number of LLL iterations of different LR algorithms with early termination in a 4×4 MIMO system with 64-QAM. 40

5.1 Column index sequence in the Incremental Sequential fcLLL algorithm using incremental sequential strategy. 42

5.2 Column index sequence in the Incremental Even-odd fcLLL algorithm using incremental even-odd strategy. 43

5.3 CCDFs of LLL iterations of different LR algorithms for 4×4 and 8×8 MIMO systems. 48

5.4 Average value and standard deviation of LLL iterations of different LR algorithms from 3×3 to 8×8 MIMO systems. 49

5.5 BER performance of dual-LR-aided MMSE detectors with different LR algorithms in an 8×8 MIMO system using 64-QAM. 51

5.6 BER performance of LR-aided MMSE-SIC detectors with different LR algorithms in an 8×8 MIMO system using 64-QAM. 52

5.7 BER performance versus number of LLL iterations of different LR algorithms in a 128×128 MIMO system using 64-QAM. 54

6.1 Proposed two-angle complex Givens rotation for column swap. 59

6.2 The proposed wordlength optimization procedure. 61

6.3 BER comparison between fixed-point and floating point of the hardware-oriented modified Incremental fcLLL algorithm in the complex LR-aided MMSE K-best detector ($K = 3$ candidates) for a 4×4 MIMO system with 64-QAM. 63

7.1 Proposed iterative architecture of the Incremental fcLLL algorithm. . 65

7.2 Architecture of the size reduction module. 67

7.3 Architecture of the Siegel condition module. 68

7.4 Architecture of the 2-angle CGR architecture for column swap. 69

7.5 The timing schedule for data operations and transferring. 73

7.6 BER comparison of LLL variants in FPGA realizations in the complex LR-aided MMSE K-best detector for a 4×4 MIMO system with 64-QAM. 75

8.1 Proposed high-level pipelined architecture of the modified Incremental fcLLL algorithm for 4×4 MIMO systems. 80

8.2 Architecture of the two-angle CGR module for column swap. The shaded area only exists in CORDIC-1/2 for vectoring mode. 81

8.3 Modules cooperation for processing $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ matrix. 82

8.4 Modules cooperation for processing $\tilde{\boldsymbol{R}}$ matrix. 83

8.5 Modules cooperation for processing $\boldsymbol{T}$ matrix. 85

8.6 Timing schedule of the whole hardware architecture. 86

8.7 Data transfer between pipelined fcLLL stages for $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ matrix. 89

8.8 Data transfer between pipelined fcLLL stages for $\tilde{\boldsymbol{R}}$ matrix. 89

8.9 Data transfer between pipelined fcLLL stages for $\boldsymbol{T}^T$ matrix. 90

8.10 BER comparison of different LR algorithms with fixed $N_{iter} = 5$ iterations in the complex LR-aided MMSE K-best detector ($K = 3$ candidates) for a 4×4 MIMO system with 64-QAM. 92

SUMMARY

Multiple-input multiple-output (MIMO) techniques are widely adopted in modern wireless systems to provide high data rate and throughput. However, appropriate MIMO detectors are needed to exploit these benefits. Among different MIMO detectors, lattice reduction (LR) aided detectors have received much interests duo to the high-performance and low-complexity property in practice. Lenstra-Lenstra-Lovász (LLL) is a widely adopted LR algorithm with desirable complexity-performance trade-off. One issue of the LLL algorithm is that the column swap may not happen in some LLL iterations, which causes inefficiency since the run-time is decided by the number of column swaps. To solve it, some greedy LLL variants have been developed so that the column swap always exists, which leads to faster convergence. However, the existing algorithms do not show how to efficiently select the LLL iterations while maintaining the performance. Another issue of the LLL algorithm is its variable complexity and run-time, which is not desirable in hardware. To solve it, some fixed-complexity LLL (fcLLL) algorithms have been proposed, which adopt a predefined column traverse strategy and iterations. However, These fcLLL algorithms are designed to process each column with equal priority, which is not optimized in terms of performance and complexity.

In this dissertation, we present enhanced greedy LLL and fcLLL algorithms for LR-aided MIMO detectors, which deal with the aforementioned shortcomings in the existing greedy LLL and fcLLL algorithms. Furthermore, we implement the proposed enhanced fcLLL algorithm in hardware by two types of architectures for low complexity and high throughput, respectively.

First, we analyze the relationship between the error performance of LR-aided MIMO detectors and the LLL algorithm by taking account of the iteration order in the LLL algorithm. The analysis acts as the theoretical foundation to design following enhanced LLL algorithms.

Second, we design enhanced greedy LLL and fcLLL algorithms. For the designed greedy LLL algorithms, we propose a relaxed Lovász condition for searching the candidate set of LLL iterations with column swap operations. Then, we propose two alternatives to select the optimal one in the candidate set of LLL iterations. Simulations show that the proposed greedy LLL algorithms not only converge faster but also exhibit much lower complexity than the existing greedy LLL variants while the error performance is maintained. For the designed fcLLL (named Incremental fcLLL) algorithms, we propose novel column traverse strategies by allocating priorities to columns based on the characteristics of LLL and MIMO detection. In addition, we propose an improved termination criterion without sacrificing the error performance in the proposed fcLLL algorithms. Simulations show that the proposed Incremental fcLLL algorithms converge faster, and yield better error performance than the existing fcLLL algorithms when the number of LLL iterations is fixed.

Third, we propose a modified Incremental fcLLL algorithm for efficient hardware implementation, which eliminates all computationally intensive operations. Then, we design a fixed-point conversion scheme to realize the fixed-point design for the modified Incremental fcLLL algorithm. The modified Incremental fcLLL algorithm and the fixed-point design facilitate the following low-complexity high-throughput hardware implementations.

Last, we concentrate on two efficient hardware architectures to implement the modified Incremental fcLLL algorithm. Our first design is a low-complexity iterative architecture, where all the modules are iteratively time-multiplexed among LLL iterations to achieve low utilization of hardware resources. The implementations on Xilinx

Virtex-4/5/7 field-programmable gate array (FPGA) devices demonstrate much lower utilization of FPGA resources while still achieving comparable throughput compared to the existing FPGA solutions. Our second design is a high-throughput pipelined architecture, where each LLL iteration corresponds to one pipelined stage to increase throughput. The implementations on Xilinx Virtex-4/5/7 FPGA devices demonstrate a processing period of 26 cycles per matrix, resulting in throughput up to 9.9 million matrices per second. Our pipelined design has much higher throughput than the existing FPGA implementations, and similar even better throughput than the recent application-specific integrated circuit (ASIC) and application-specific instruction set processor (ASIP) implementations.

CHAPTER I

INTRODUCTION

1.1 Motivations

Multiple-input multiple-output (MIMO) techniques [43, 47, 90] have been adopted in modern wireless standards (e.g., IEEE 802.11n/ac, 3GPP LTE/LTE-A) due to the high spectral efficiency and enhanced coverage. However, appropriate MIMO detectors are needed to exploit these benefits. The optimal detector for MIMO systems is the maximum likelihood (ML) detector, but it exhibits exponential complexity [27]. To alleviate the complexity of ML detector, linear detectors (LDs), successive-interference-cancellation (SIC) detectors, and K-best detectors are adopted but with inferior performance due to diversity loss [65].

MIMO detection can also be formulated as the closest vector problem (CVP) by means of lattice theory [1]. Although the CVP can be solved exactly by using sphere decoding with lower complexity than the ML detector, its complexity is still exponential with respect to the number of transmit antennas [22] and the randomness of the complexity leads to inefficient hardware implementation. To further reduce the complexity while keeping high performance in MIMO detection, lattice reduction (LR) algorithms have been proposed and gained considerable attentions [25, 38, 40, 80, 86, 91, 93, 95, 98, 103]. Among them, Lenstra-Lenstra-Lovász (LLL) algorithm [29] and its complex-valued versions [17, 41, 44] are widely adopted due to the high performance and polynomial complexity in average [24]. It has been shown that the LLL-aided LDs and SIC detectors can collect full receive diversity as the ML detector [17, 41, 63, 84]. Furthermore, the LR-aided K-best detector can achieve near-ML performance [54, 79, 101]. When LR-aided K-best detector combined with minimum

mean-square error (MMSE) regularization in the complex domain (i.e., complex LR-aided MMSE K-best detector)[79], it brings almost the same performance as the ML detector with small number of candidates. Another form of LLL technique, named dual LLL (DLLL) algorithm, which performs the LLL algorithm in the dual space [33, 35], outperforms the LLL algorithm in LR-aided LDs in terms of error performance [32, 96] without increasing complexity [85]. But the DLLL exhibits higher complexity while comparable error performance compared to the LLL in LR-aided SIC detectors [85]. Another common LR scheme is Seysen's algorithm (SA)[51, 53, 92], which minimizes Seysen's metric so that both the primal and dual bases are simultaneously reduced. The SA has higher complexity than the LLL/DLLL, but it exhibits similar error performance as the DLLL in LR-aided LDs [85] and similar error performance as the LLL in LR-aided SIC detectors [6, 85]. Therefore, we mainly focus on DLLL-aided LDs and LLL-aided SIC detectors in this dissertation for high performance and low complexity.

One issue of the LLL algorithm is that the column swap may not happen in some LLL iterations. This causes some inefficiency since the number of LLL iterations or the run-time is decided by the number of column swaps [29]. To solve it, some greedy LLL variants [71, 97, 99] have been developed so that the column swap always exists in each LLL iteration, which leads to faster convergence than the original LLL algorithm. However, the existing algorithms do not show how to efficiently select the LLL iterations while maintaining the performance. Another issue of the LLL algorithm is the variable complexi and non-deterministic iteration order, which is not desirable for hardware implementation. To solve it, some fixed-complexity LLL (fcLLL) algorithms with predefined number of iterations and traversal order among iterations have been proposed in [66, 67]. However, These fcLLL algorithms are designed to process each column with equal priority, which is not optimized in terms of performance and complexity.

Besides extensive theoretical researches, LR algorithms also have attracted many hardware implementations [2, 5, 9, 18, 19, 30, 55–57, 60, 69, 104] by field-programmable gate array (FPGA), application-specific integrated circuit (ASIC), and application-specific instruction set processor (ASIP) devices. Among these hardware realizations, most of them are based on LLL or fcLLL algorithms due to the performance-complexity tradeoff. The original LLL algorithm is not desirable for direct hardware realization because of the nondeterministic iteration number and order. So the existing implementations either fix the (maximum) iteration number as in [5, 9, 18, 19, 56, 69], or fix both iteration number and order as in [30, 55] by using fcLLL algorithms. It has been shown that our proposed Incremental fcLLL requires less number of iterations than other fcLLL algorithms to obtain the best-achievable performance in LR-aided MIMO detectors [72], which implies the potential of higher throughput with lower complexity in hardware. However, the hardware realization based on the Incremental fcLLL has not been thoroughly investigated.

1.2 Objectives

The objective of the proposed research is to design low-complexity and high-performance enhanced LLL algorithms for LR-aided MIMO detectors, in both theory and hardware implementation. To be specific, the goals are given as follows:

1. Propose enhanced greedy LLL and fcLLL algorithms with much faster convergence and lower complexity by taking account of the shortcomings of the existing greedy LLL and fcLLL algorithms;

2. Modify the proposed fcLLL algorithms and develop the corresponding fixed-point design for efficient hardware implementation;

3. Develop low-complexity high-throughput hardware architectures of the modified fcLLL algorithm.

1.3 Outline

The rest of the dissertation is organized as follows:

Chapter 2 gives a brief introduction to LR-aided MIMO detection and the widely adopted LLL variants, followed by existing hardware implantations of lattice reduction algorithms.

Chapter 3 analyzes the error performance of the LR-aided MIMO detectors, which is the major theoretical principle to design enhanced LLL algorithms in Chapters 4 and 5.

Chapter 4 proposes two enhanced greedy LLL algorithms with faster convergence and much lower complexity than the existing greedy LLL variants.

Chapter 5 proposes Incremental fcLLL algorithms with novel column traverse strategies, which not only converge faster but also exist much lower complexity than existing fcLLL algorithms.

Chapter 6 proposes a hardware-oriented modified Incremental fcLLL algorithm and develops the corresponding fixed-point design, which facilitate the efficient hardware implementations in Chapters 7 and 8.

Chapter 7 develops a low-complexity iterative architecture to implement the proposed hardware-oriented modified Incremental fcLLL algorithm.

Chapter 8 develops a high-throughput pipelined architecture to implement the proposed hardware-oriented modified Incremental fcLLL algorithm.

Chapter 9 concludes the dissertation and provides some future research suggestions.

1.4 Notations

Boldface upper- and lower-case letters represent matrices and column vectors, respectively. $A_{i,k}$ denotes the (i,k)th entry of matrix $\boldsymbol{A}$. $\boldsymbol{A}_{a:b,c:d}$ denotes a submatrix of $\boldsymbol{A}$ including entries from rows a to b and from columns c to d. If only : is used,

that corresponds to entries in a complete row or column. Index notation represented as $n = a : b : c$ means all the index numbers from a to c with step length as b. The msb represents most significant bit, and $>>$ denotes right shift. $\boldsymbol{I}_N$ denotes the $N \times N$ identity matrix, $\mathbf{1}_{N\times 1}$ is the $N \times 1$ vector of ones, and $\mathbf{0}_{N\times 1}$ is the $N \times 1$ vector of zeros. The conjugate, transpose, and Hermitian transpose are denoted by $(\cdot)^*$, $(\cdot)^T$, and $(\cdot)^{\mathcal{H}}$, respectively. The real and imaginary parts of a complex number are represented as $\mathfrak{R}[\cdot]$ and $\mathfrak{I}[\cdot]$, $j = \sqrt{-1}$, and $\mathbb{Z}$ is the integer set. The inner product of two vectors $\boldsymbol{u}$, $\boldsymbol{v}$ is denoted by $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$. $\lfloor a \rceil$ represents rounding to the nearest integer of a, $\| \cdot \|$ denotes the 2-norm, and $| \cdot |$ is the absolute value of a scalar.

CHAPTER II

BACKGROUND

2.1 MIMO Detection

2.1.1 Conventional MIMO Detection

We consider a generalized flat-fading MIMO system [43, 47] with N_t transmit and N_r receive antennas. The received signal vector $\boldsymbol{y} = [y_1, y_2, \cdots, y_{N_r}]^T$ can be expressed as

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{s} + \boldsymbol{w}, \tag{2.1}$$

where $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_{N_t}]$ is an $N_r \times N_t$ ($N_r \geq N_t$) channel matrix which consists of independent and identically distributed (i.i.d.) complex Gaussian entries with zero mean and unit variance, $\boldsymbol{s} = [s_1, s_2, \cdots, s_{N_t}]^T \in \mathcal{S}^{N_t}$ is the transmitted signal vector drawn from the QAM constellation set $\mathcal{S}$ with $E[\boldsymbol{s}\boldsymbol{s}^{\mathcal{H}}] = \sigma_s^2 \boldsymbol{I}_{N_t}$, and $\boldsymbol{w} = [w_1, w_2, \cdots, w_{N_r}]^T$ is the additive white Gaussian noise vector with zero mean and covariance matrix $\sigma_w^2 \boldsymbol{I}_{N_r}$. The real and imaginary parts of $\boldsymbol{s}$ are integers from the set $\{-\sqrt{\mathcal{M}}+1, \ldots, -1, 1, \ldots, \sqrt{\mathcal{M}}-1\}$ with $\mathcal{M}$ being the constellation size of $\mathcal{S}$. We assume that $\boldsymbol{H}$ is known at the receiver side but unknown at the transmitter side.

Based on the model in (2.1), the ML detector [27] is given as

$$\hat{\boldsymbol{s}} = \arg \min_{\boldsymbol{s} \in \mathcal{S}^{N_t}} \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{s}\|^2. \tag{2.2}$$

Although ML detector is optimal in terms of error performance, it exhibits exponential complexity even through solved by the efficient sphere decoding [22, 27].

For the low-complexity LD with respect to zero forcing (ZF) criterion, it is expressed as

$$\hat{\boldsymbol{s}} = \mathcal{Q}(\boldsymbol{H}^{\dagger}\boldsymbol{y}), \tag{2.3}$$

where $\boldsymbol{H}^{\dagger} = (\boldsymbol{H}^{\mathcal{H}}\boldsymbol{H})^{-1}\boldsymbol{H}^{\mathcal{H}}$ is the Moore-Penrose pseudo inverse of $\boldsymbol{H}$ and $\mathcal{Q}(\cdot)$ is the symbol-wise quantizer to the nearest point in the constellation set $\mathcal{S}$.

For the SIC detector, QR decomposition $\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{R}$ can be firstly performed, where $\boldsymbol{Q}^{\mathcal{H}}\boldsymbol{Q} = \boldsymbol{I}$ and $\boldsymbol{R}$ is an upper triangular matrix. Then, multiplying $\boldsymbol{Q}^{\mathcal{H}}$ on both sides of (2.1), we get

$$\breve{\boldsymbol{y}} = \boldsymbol{R}\boldsymbol{s} + \boldsymbol{n}, \tag{2.4}$$

where $\breve{\boldsymbol{y}} = \boldsymbol{Q}^{\mathcal{H}}\boldsymbol{y}$ and $\boldsymbol{n} = \boldsymbol{Q}^{\mathcal{H}}\boldsymbol{w}$. Finally, the detection is operated sequentially from the N_tth symbol to the 1st symbol by exploiting the characteristics of $\boldsymbol{R}$ as follows

$$\hat{s}_i = \mathcal{Q}\left(\frac{\breve{y}_i - \sum_{n=i+1}^{N_t} R_{i,n}\hat{s}_n}{R_{i,i}}\right), \quad \text{for} \quad i = N_t, N_t - 1, \cdots, 1. \tag{2.5}$$

For the K-best detector, we can rewrite the problem in (2.2) by combining (2.4) as follows

$$\hat{\boldsymbol{s}} = \arg\min_{\boldsymbol{s}\in\mathcal{S}^{N_t}} \|\breve{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{s}\|^2 = \arg\min_{\boldsymbol{s}\in\mathcal{S}^{N_t}} \sum_{i=1}^{N_t} |\breve{y}_i - \sum_{j=i}^{N_t} R_{i,j}s_j|^2. \tag{2.6}$$

Then, the detection is operated sequentially from the $n = N_t$th level to the $n = 1$st level, where only K candidates with smallest partial cost would be survived at each level of the decoding tree. The partial cost of a partial candidate $\mathbf{s}_k^{(n)} = [s_{k,n}^{(n)}, \cdots, s_{k,N_t}^{(n)}]^T$ at the nth level is defined as

$$cost_k^{(n)} = \sum_{l=n}^{N_t} |\breve{y}_l - \sum_{j=l}^{N_t} R_{l,j}z_{k,j}^{(n)}|^2, \quad k = 1, 2, ..., K. \tag{2.7}$$

At the nth layer, the algorithm finds the K best partial candidates $[\mathbf{s}_1^{(n)}, \mathbf{s}_2^{(n)}, \cdots, \mathbf{s}_K^{(n)}]$, i.e., the K partial candidates that have the minimum costs among all the children of the K partial candidates $[\mathbf{s}_1^{(n+1)}, \mathbf{s}_2^{(n+1)}, \cdots, \mathbf{s}_K^{(n+1)}]$ in the previous $(n+1)$st layer. After the 1st layer is reached, the final solution is the candidate with minimum cost among the K best candidates as

$$\hat{\boldsymbol{s}} = \arg\min_{\boldsymbol{s}\in\{\boldsymbol{s}_k\}_{k=1}^{K}} \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{s}\|^2, \tag{2.8}$$

For minimum mean squared error (MMSE) regularized LDs and SIC/K-best detectors, they have the same formulation as (2.3) and (2.5) by using the following extended system [45, 79, 83]

$$\bar{\boldsymbol{H}} = \begin{bmatrix} \boldsymbol{H} \\ \frac{\sigma_w}{\sigma_s}\boldsymbol{I}_{N_t} \end{bmatrix}, \quad \bar{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{0}_{N_t \times 1} \end{bmatrix}. \tag{2.9}$$

2.1.2 Lattice-Reduction-Aided MIMO Detectors

Given the channel matrix $\boldsymbol{H}$, we can obtain a complex lattice which is defined as

$$\mathcal{L}(\boldsymbol{H}) = \left\{ \boldsymbol{v} \,\middle|\, \boldsymbol{v} = \sum_{i=1}^{N_t} c_i \boldsymbol{h}_i,\ c_i \in \mathbb{Z}_j \right\}, \tag{2.10}$$

where $\mathbb{Z}_j = \{a + bj | a, b \in \mathbb{Z}\}$ represents the Gaussian integer ring, and the column vectors $\boldsymbol{h}_i$ of the matrix $\boldsymbol{H}$ represent a basis of the lattice. Given a basis $\boldsymbol{H}$, LR algorithms produce a near-orthogonal basis as $\tilde{\boldsymbol{H}} = \boldsymbol{HT}$ ($\boldsymbol{T}$ is a unimodular matrix consisted of Gaussian integers with determinant ± 1 or $\pm j$). With this near-orthogonal $\tilde{\boldsymbol{H}}$, much better error performance can be obtained in the MIMO systems with the low-complexity LDs or SIC/K-best detectors.

To incorporate LR into MIMO detection, we rewrite the system model by applying scaling and shifting on $\boldsymbol{s}$ such that $\boldsymbol{Hs}$ can be viewed as a lattice point with $\boldsymbol{H}$ as basis, i.e.,

$$\boldsymbol{y}' = \frac{\boldsymbol{y} + \boldsymbol{H}\boldsymbol{1}(1+j)}{2} = \boldsymbol{HT}\boldsymbol{T}^{-1}\frac{\boldsymbol{s} + \boldsymbol{1}(1+j)}{2} + \frac{1}{2}\boldsymbol{w} = \tilde{\boldsymbol{H}}\boldsymbol{z} + \frac{1}{2}\boldsymbol{w}, \tag{2.11}$$

where $\boldsymbol{1}$ is the $N_t \times 1$ vector of ones, $\tilde{\boldsymbol{H}} = \boldsymbol{HT}$ is the reduced basis, and $\boldsymbol{z}$ is the vector in the lattice-reduced domain. Then, the low-complexity LDs or SIC/K-best detectors are performed according to (2.11) to obtain the estimated $\boldsymbol{z}$ as $\hat{\boldsymbol{z}}$. Finally, the estimation of the transmitted symbols $\boldsymbol{s}$ are computed as

$$\hat{\boldsymbol{s}} = \mathcal{Q}\left[2\boldsymbol{T}\hat{\boldsymbol{z}} - \boldsymbol{1}(1+j)\right], \tag{2.12}$$

where $\mathcal{Q}(\cdot)$ is the symbol-wise quantizer to the nearest point in the constellation set.

The aforementioned LR algorithm is based on the primal basis. We can also reduce the dual basis with LR algorithms. The dual lattice $\mathcal{L}^\star$ of a primal lattice $\mathcal{L}$ is defined as

$$\mathcal{L}^\star = \{\boldsymbol{u} \,|\langle \boldsymbol{u}, \boldsymbol{v}\rangle \in \mathbb{Z}_j,\ \forall\, \boldsymbol{v} \in \mathcal{L}\}\,. \tag{2.13}$$

Therefore, the dual basis $\boldsymbol{H}^\star$ associated with the primal basis $\boldsymbol{H}$ can be defined as [26, 31]

$$\boldsymbol{H}^\star = [\boldsymbol{h}_1^\star, \boldsymbol{h}_2^\star, \ldots, \boldsymbol{h}_{N_t}^\star] = (\boldsymbol{H}^\dagger)^{\mathcal{H}}\boldsymbol{J} = \boldsymbol{H}(\boldsymbol{H}^{\mathcal{H}}\boldsymbol{H})^{-1}\boldsymbol{J}, \tag{2.14}$$

where $\boldsymbol{J}$ is the column-reversing matrix with anti-diagonal entries as ones and others as zeros, and satisfies $\boldsymbol{J}\boldsymbol{J} = \boldsymbol{I}$. For the dual basis $\boldsymbol{H}^\star$ of the primal basis $\boldsymbol{H}$, LR algorithms produce a reduced dual basis as $\tilde{\boldsymbol{H}}^\star = \boldsymbol{H}^\star\boldsymbol{T}^\star$. The reduced dual basis and primal basis have the relationship $\tilde{\boldsymbol{H}}^\star = (\tilde{\boldsymbol{H}}^\dagger)^{\mathcal{H}}\boldsymbol{J}$ and $\boldsymbol{T}^\star = \boldsymbol{J}(\boldsymbol{T}^{-1})^{\mathcal{H}}\boldsymbol{J}$ based on (2.14).

For the dual-LR-aided LDs, by denoting the reduced dual channel matrix as $\tilde{\boldsymbol{H}}^\star = \boldsymbol{H}^\star\boldsymbol{T}^\star$, the estimation of $\boldsymbol{z}$ is calculated as

$$\hat{\boldsymbol{z}} = \lfloor \tilde{\boldsymbol{H}}^\dagger \boldsymbol{y}' \rceil = \lfloor (\tilde{\boldsymbol{H}}^\star\boldsymbol{J})^{\mathcal{H}}\boldsymbol{y}' \rceil. \tag{2.15}$$

From the estimation of $\boldsymbol{z}$, the symbols in s-domain can be mapped as

$$\hat{\boldsymbol{s}} = \mathcal{Q}\left[2\boldsymbol{T}\hat{\boldsymbol{z}} - (1+j)\mathbf{1}_{N_t\times 1}\right]. \tag{2.16}$$

For the LR-aided SIC detector, after multiplying $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ at both sides of (2.11) with $\tilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{T} = \tilde{\boldsymbol{Q}}\tilde{\boldsymbol{R}}$, the system model in (2.11) can be represented as

$$\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{R}}\boldsymbol{z} + \tilde{\boldsymbol{n}}, \tag{2.17}$$

where $\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{Q}}^{\mathcal{H}}\boldsymbol{y}'$, and $\tilde{\boldsymbol{n}} = (\tilde{\boldsymbol{Q}}^{\mathcal{H}}\boldsymbol{w})/2$. It can be seen that (2.17) has the same form as the one in (2.4). Therefore, the SIC estimation can be firstly performed in z-domain, using (2.5) with the $\mathcal{Q}$ quantizer replaced by the $\lfloor\cdot\rceil$ integer rounding operation, to obtain $\hat{\boldsymbol{z}}$. Then, the final estimation of the transmitted vector in the s-domain can be obtained by using (2.16).

Similarly, for the LR-aided K-best detector, the detection is firstly performed in z-domain based on (2.17) from the $n = N_t$th level to the $n = 1$st level. After the 1st layer is reached, the K best candidates with minimum costs can be obtained as

$$\hat{\boldsymbol{s}}_k = \mathcal{Q}(2\boldsymbol{T}\boldsymbol{z}_k^{(1)} - (1+j)\mathbf{1}_{N_t\times 1}), \qquad k = 1, 2, ..., K. \tag{2.18}$$

Then, the final result in the s-domain can be obtained by

$$\hat{\boldsymbol{s}} = \arg \min_{\tilde{\boldsymbol{s}}\in\{\hat{\boldsymbol{s}}_k\}_{k=1}^{K}} \|\boldsymbol{y} - \boldsymbol{H}\tilde{\boldsymbol{s}}\|^2. \tag{2.19}$$

Note that the aforementioned dual-LR-aided LDs and LR-aided SIC/K-best detectors are based on the ZF estimation criteria, which is generally not optimal in terms of diversity-multiplexing tradeoff (DMT) [23, 62]. To achieve the DMT optimality, we can extend the aforementioned formulations with MMSE regularization criterion [41, 79, 84] by adopting the formulations (2.9).

2.2 LLL Lattice Reduction Algorithms

2.2.1 LLL and ELLL Algorithms

The LLL algorithm produces a near to orthogonal basis called LLL-reduced basis. The following is the mathematical definition.

Definition 2.1 (LLL-reduced basis [41]): Let $\tilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{T} = \tilde{\boldsymbol{Q}}\tilde{\boldsymbol{R}}$ be the QR decomposition of the transformed channel matrix $\tilde{\boldsymbol{H}}$. $\tilde{\boldsymbol{H}}$ is called an LLL-reduced basis if the following two conditions are satisfied:

$$|\Re[\tilde{R}_{k,i}]| \le \frac{1}{2}|\tilde{R}_{k,k}|, \ |\Im[\tilde{R}_{k,i}]| \le \frac{1}{2}|\tilde{R}_{k,k}|, \ \forall\ 1 \le k < i \le N_t, \tag{2.20}$$

$$\delta|\tilde{R}_{k-1,k-1}|^2 \le |\tilde{R}_{k,k}|^2 + |\tilde{R}_{k-1,k}|^2, \ \ \forall\ 2 \le k \le N_t, \tag{2.21}$$

where (2.20) is called size reduction condition, (2.21) is Lovász condition, and $1/2 < \delta \le 1$ is a quality parameter selected to control the performance-complexity tradeoff (larger δ leads to better performance with higher complexity).

Another LLL variant is called effective LLL (ELLL) [21, 34] which produces an ELLL-reduced basis by fixing the index i as $i = k+1$ in (2.20). We denote it as effective size reduction condition in the ELLL algorithm. The ELLL algorithm exhibits less complexity than the LLL algorithm but maintains the same error performance in LR-aided SIC detection as proved in [18]. For LR-aided K-best detection, similar result can be achieved as the following proposition states:

Proposition 2.1 *ELLL and LLL have the same error performance in LR-aided K-best detection.*

Proof See Appendix A.

The detailed complex-valued LLL/ELLL algorithm based on QR preprocessing is summarized in Table 2.1, where the preprocessing part can also be sorted QR decomposition (SQRD) or MMSE-SQRD [84] to reduce LLL/ELLL's iterations. The LLL/ELLL algorithm repeatedly performs three steps, i.e., *(effective) size reduction*, *Lovász condition evaluation*, and *column swap* (if the Lovász condition is not satisfied), to produce an LLL/ELLL-reduced basis. To compare the convergence among LLL variants later, we use the following definition of LLL iteration.

Definition 2.2 (LLL iteration): One LLL iteration is defined as one-time sequential execution of (effective) size reduction, Lovász condition evaluation, and possible column swap in the LLL algorithm or its variants.

2.2.2 Greedy and Fixed-Complexity LLL Algorithms

2.2.2.1 Greedy LLL Algorithms

One issue of the LLL/ELLL algorithm is that the column swap does not happen if the Lovász condition is satisfied in an LLL iteration. This slows down the convergence. To understand the LLL's convergence, we introduce the definition of LLL potential.

Table 2.1: The LLL/ELLL Algorithm

Input: $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{P}$ (after QR/SQRD/MMSE-SQRD†)
Output: $\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{R}}, \boldsymbol{T}$

1: Initialize: $\tilde{\boldsymbol{Q}} = \boldsymbol{Q},\ \tilde{\boldsymbol{R}} = \boldsymbol{R},\ \boldsymbol{T} = \boldsymbol{P},\ \delta \in (1/2, 1]$
2: $k = 2$
3: **while** $k \leq N_t$
4: **for** $n = k-1 : -1 : 1$(LLL) or **for** $n = k-1$(ELLL)
5: $u = \lceil \tilde{R}_{n,k} / \tilde{R}_{n,n} \rfloor$
6: **if** $u \neq 0$
7: $\tilde{\boldsymbol{R}}_{1:n,k} = \tilde{\boldsymbol{R}}_{1:n,k} - u\tilde{\boldsymbol{R}}_{1:n,n}$
8: $\boldsymbol{T}_{:,k} = \boldsymbol{T}_{:,k} - u\boldsymbol{T}_{:,n}$
9: **end**
10: **end** } ***(effective) size reduction***
11: **if** $\delta|\tilde{R}_{k-1,k-1}|^2 > |\tilde{R}_{k,k}|^2 + |\tilde{R}_{k-1,k}|^2$ } ***Lovasz condition***
12: swap columns $k-1$ and k in $\tilde{\boldsymbol{R}}$ and $\boldsymbol{T}$
13: $\boldsymbol{\Theta} = \begin{bmatrix} \alpha^* & \beta \\ -\beta & \alpha \end{bmatrix}$ with $\alpha = \frac{\tilde{R}_{k-1,k-1}}{||\tilde{\boldsymbol{R}}_{k-1:k,k-1}||}$, $\beta = \frac{\tilde{R}_{k,k-1}}{||\tilde{\boldsymbol{R}}_{k-1:k,k-1}||}$
14: $\tilde{\boldsymbol{R}}_{k-1:k,k-1:N_t} = \boldsymbol{\Theta}\tilde{\boldsymbol{R}}_{k-1:k,k-1:N_t}$
15: $\tilde{\boldsymbol{Q}}_{:,k-1:k} = \tilde{\boldsymbol{Q}}_{:,k-1:k}\boldsymbol{\Theta}^{\mathcal{H}}$ } ***column swap***
16: $k = \max(k-1, 2)$
17: **else**
18: $k = k + 1$
19: **end**
20: **end**

†QR: $\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{R}, \boldsymbol{P} = \boldsymbol{I}$; SQRD: $\boldsymbol{H}\boldsymbol{P} = \boldsymbol{Q}\boldsymbol{R}$; MMSE-SQRD: $\bar{\boldsymbol{H}}\boldsymbol{P} = [\boldsymbol{Q}^T \hat{\boldsymbol{Q}}^T]^T \boldsymbol{R}$.
$\boldsymbol{Q}$ is an $N_r \times N_t$ matrix, $\boldsymbol{R}$ and $\boldsymbol{P}$ are $N_t \times N_t$ matrices.

Definition 2.3 (LLL Potential [29, 34]): A positive number D is called LLL potential as

$$D \stackrel{\text{def}}{=} \prod_{i=1}^{N_t-1} d_i = \prod_{i=1}^{N_t-1} |\tilde{R}_{i,i}|^{2(N_t-i)}, \tag{2.22}$$

where $d_i = \det^2(\mathcal{L}_i) = \prod_{m=1}^{i} |\tilde{R}_{m,m}|^2$ and $\mathcal{L}_i$ is the sub-lattice spanned by column vectors $\tilde{\boldsymbol{q}}_1, \ldots, \tilde{\boldsymbol{q}}_i$ of matrix $\tilde{\boldsymbol{Q}}$ from $\tilde{\boldsymbol{H}} = \tilde{\boldsymbol{Q}}\tilde{\boldsymbol{R}}$.

The size reduction does not change the value of LLL Potential D, since the diagonal elements in $\tilde{\boldsymbol{R}}$ are unchanged. The value of D only changes after a column swap. It has been shown that D is monotonically decreasing during the execution of the

LLL algorithm and there is a lower bound for D that depends on $\mathcal{L}(\boldsymbol{H})$ [29]. Hence, the LLL algorithms would terminate after a finite number of LLL iterations.

The existing greedy LLL algorithms, i.e., the possible-swap LLL with optimal swap selection criterion (PSLLL-OSSC) in [99] and the greedy diagonal reduction (GDR) in [97], take the convergence characteristic of the LLL algorithm into consideration, which guarantee that the column swap is performed in each LLL iteration for faster convergence. To be specific, both algorithms add two major modifications to the original LLL algorithm before each LLL iteration: one is to find the candidate set of LLL iterations with column swap operations; the other is to select an LLL iteration in the candidate set such that the decrescence of LLL potential is maximized each time. These two modifications make the existing greedy LLL algorithms converge faster than the original LLL algorithm. Fig. 2.1 depicts the empirical complementary cumulative distribution functions (CCDFs) of the LLL iterations and column swaps for different LR algorithms. It can be seen that in the original LLL/ELLL algorithms the number of column swaps is much less than the number of LLL iterations; while greedy LLL algorithms improve the convergence speed such that the number of column swaps equals the number of the LLL iterations.

2.2.2.2 Fixed-Complexity LLL Algorithms

In practice, one issue of the LLL/ELLL algorithm is that the column traverse structure, i.e., the sequence of the column index k, is not deterministic, since the k can be increased or decreased during LLL execution (lines 16, 18 of Table 2.1) depending on whether Lovász condition is satisfied or not. Another issue of the LLL algorithm is that the number of LLL iterations is variable, which can even be infinite in some worst cases [24]. Due to these issues, it is not desirable to directly implement the LLL algorithm in hardware.

To solve the aforementioned issues in the LLL algorithm, an fcLLL algorithm is

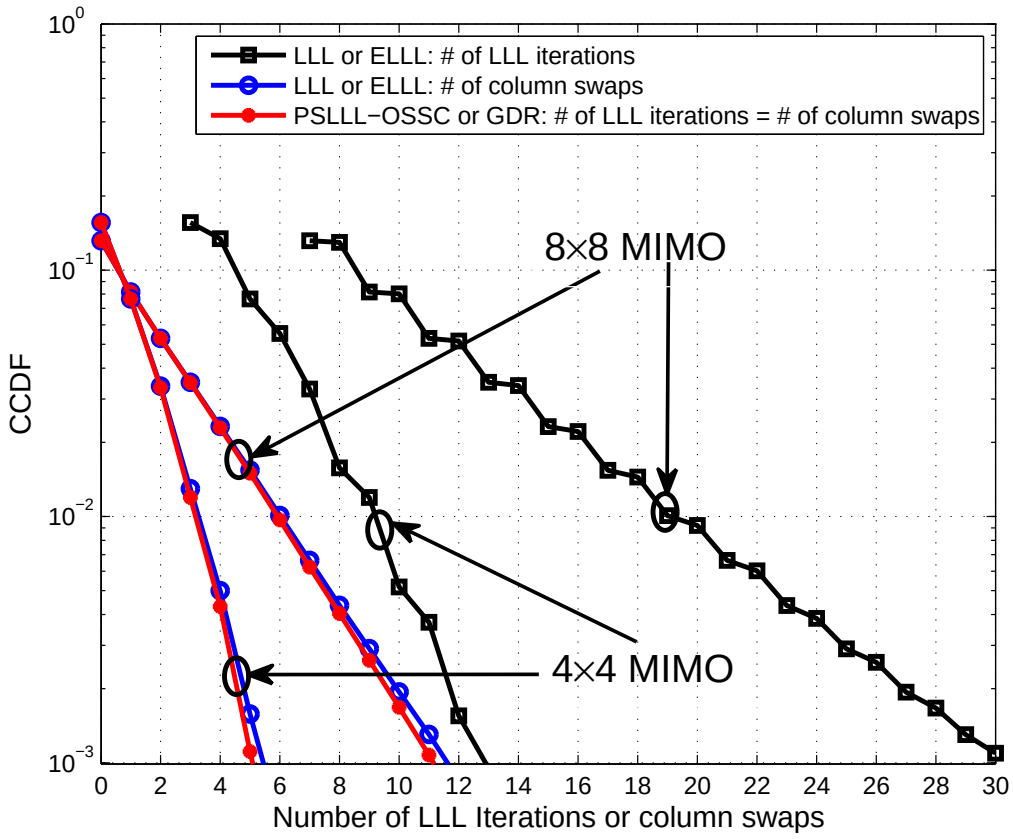


Figure 2.1: Convergence comparisons of different LR algorithms in 4×4 and 8×8 MIMO systems. For each LR algorithm, MMSE-SQRD is adopted, the parameter $\delta = 0.75$, and 10^6 i.i.d. Gaussian channel realizations are simulated.

proposed in [66], which has three modifications compared to the LLL algorithm as follows:

1) Deterministic Column Traverse Strategy: The fcLLL uses a fixed-structure column traverse strategy, where the column index repeats a super-iteration that monotonically increments from 2 to N_t.

2) Novel Termination Criterion: The fcLLL uses a flag to track *column swap*. As soon as no *column swap* happens in a super-iteration, the fcLLL terminates with an LLL-reduced basis.

3) Fixed (Maximum) Number of LLL Iterations: The fcLLL either adopts a fixed

number of LLL iterations to obtain a constant throughput, or adopts early termination mechanism that uses an upper bound to the number of LLL iterations to guarantee throughput in the worst case.

Due to the sequential column traverse strategy, we denote this fcLLL as "Sequential fcLLL". Note that the three modifications in Sequential fcLLL can be also used in the effective LLL algorithm for LR-aided SIC detection as in [36].

Another LLL variant in [20, 67] also repeatedly runs a super-iteration consisted of even numbers increased from 2 to N_t followed by odd numbers increased from 3 to N_t, and has the same termination criterion as Sequential fcLLL. We can simply add fixed (maximum) number of LLL iterations to this variant to obtain another fcLLL algorithm, which is denoted as "Even-odd fcLLL" here based on its characteristic of the column traverse strategy.

Because of the similarity of the Sequential fcLLL and the Even-odd fcLLL algorithms, we summarize a unified form in Table 2.2. In this table, one super-iteration corresponds to lines 6-14 while one LLL iteration corresponds to lines 7-13, N_{max} is the maximum number of LLL iterations, and *stop* flag is used to end the algorithms when no *column swap* happens in a super-iteration. Alternatively, we can fix the number of LLL iterations to get constant throughput by removing the *stop* flag in lines 3, 5, and 10, replacing line 4 by "**for** $n_{iter} = 1 : 1 : N_{max}$", and deleting lines 13. Note that our designed N_{max} in Table 2.2 can be any integer number larger than zero instead of multiple super-iterations as $N_{max} = Y(N_t - 1), Y \geq 1$ in [66]. This is more flexible for practical implementation.

2.3 Existing Hardware Implementations of Lattice Reduction Algorithms

Besides the extensive theoretical research interests, LR algorithms for MIMO detection have also attracted many hardware implementations since 2007 [10]. Among

Table 2.2: The Unified Form of Sequential FcLLL and Even-odd FcLLL

Input: $\bar{\boldsymbol{Q}}, \bar{\boldsymbol{R}}$
Output: $\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{R}}, \boldsymbol{T}$

1: Initialize: $\tilde{\boldsymbol{Q}} = \boldsymbol{Q}, \tilde{\boldsymbol{R}} = \boldsymbol{R}, \boldsymbol{T} = \boldsymbol{I}, \delta \in (1/2, 1], N_{max}$
2: $n_{iter} = 1$
3: $stop$ = FALSE
4: **while** $stop$ = FALSE
5: stop = TRUE
6: **for** $k = \begin{cases} 2:1:N_t & \text{if Sequential fcLLL} \\ 2:2:N_t,\ 3:2:N_t & \text{if Even-odd fcLLL} \end{cases}$
7: Execute ***size reduction***
8: **if** $\delta|\tilde{R}_{k-1,k-1}|^2 > |\tilde{R}_{k,k}|^2 + |\tilde{R}_{k-1,k}|^2$
9: Execute ***column swap***
10: $stop$ = FALSE
11: **end**
12: $n_{iter} = n_{iter} + 1$
13: **if** $n_{iter} > N_{max}$ **return end**
14: **end**
15: **end**

these implementations, small parts of them are based on SA algorithms as in [52, 82], while most parts of them are mainly based on LLL and fcLLL algorithms. Since the original LLL algorithm is not desirable for direct hardware realization due to the nondeterministic iteration number and order, the existing LLL based implementations usually fix the (maximum) iteration number as in [5, 9, 18, 19, 56, 69]. The fcLLL based implementations fix both iteration number and iteration order to facilitate hardware realization [2, 30, 55]. Based on the characteristics of the iteration sequence, the existing fcLLLs can be categorized as: the Sequential fcLLL [66] and its hardware implementation [2, 55], the Even-odd fcLLLL [67] and its hardware implementation [30]. As will be discussed in Chapter 5, we propose another category of fcLLL algorithm named Incremental fcLLL, which requires less number of iterations than other fcLLLs to obtain the best-achievable performance in LR-aided detectors [72]. It implies the potential of higher throughput with lower complexity in hardware, and this potential will be fully explored in our implementations in Chapters 7 and 8.

CHAPTER III

PERFORMANCE ANALYSIS OF LR-AIDED MIMO DETECTORS

In this chapter, we consider the relationship between the error performance of LR-aided MIMO detectors and the LLL LR algorithms [74], which is the major theoretical principle to design enhanced LLL algorithms. In the following sections, we first analyze the error performance of DLLL-aided LDs, and then investigate the error performance of LLL-aided SIC/K-best detectors. The contents of this chapter are based on our publication [74].

3.1 Error Performance Analysis of DLLL-aided LDs

For the DLLL-aided LDs, based on (2.15), the output in z-domain can be written as

$$\hat{\boldsymbol{z}} = \boldsymbol{z} + \lfloor \boldsymbol{n} \rceil, \tag{3.1}$$

where $\boldsymbol{n} = \tilde{\boldsymbol{H}}^{\dagger}\boldsymbol{w}/2 = (\tilde{\boldsymbol{H}}^{\star}\boldsymbol{J})^{\mathcal{H}}\boldsymbol{w}/2 = (\tilde{\boldsymbol{Q}}^{\star}\tilde{\boldsymbol{R}}^{\star}\boldsymbol{J})^{\mathcal{H}}\boldsymbol{w}/2$ is complex Gaussian distributed with zero mean and covariance matrix as

$$E[\boldsymbol{n}\boldsymbol{n}^{\mathcal{H}}] = \frac{1}{4}\sigma_w^2\tilde{\boldsymbol{H}}^{\dagger}(\tilde{\boldsymbol{H}}^{\dagger})^{\mathcal{H}} = \frac{1}{4}\sigma_w^2\boldsymbol{J}(\tilde{\boldsymbol{R}}^{\star})^{\mathcal{H}}\tilde{\boldsymbol{R}}^{\star}\boldsymbol{J}. \tag{3.2}$$

Let us define $\boldsymbol{C} = \boldsymbol{J}(\tilde{\boldsymbol{R}}^{\star})^{\mathcal{H}}\tilde{\boldsymbol{R}}^{\star}\boldsymbol{J}$ and $e_i = z_i - \hat{z}_i$. Then, following the similar procedure as in [41, 102], we obtain the pairwise error probability (PEP), i.e., the probability that z_i is erroneously detected as $\hat{z}_i \neq z_i$ given the reduced channel $\tilde{\boldsymbol{H}}$ as

$$P(z_i \to \hat{z}_i|\tilde{\boldsymbol{H}}) = Q\left(\sqrt{\frac{2|e_i|^2}{\sigma_w^2 C_{i,i}}}\right), \tag{3.3}$$

where $Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^{+\infty} e^{-\frac{t^2}{2}}\,dt$ and $C_{i,i}$ denotes the ith diagonal element of $\boldsymbol{C}$. Furthermore, we find the relationship between $\boldsymbol{C}$ and the dual basis after lattice reduction

as

$$C_{i,i} = ||\tilde{\boldsymbol{h}}_k^\star||^2 = \sum_{\ell=1}^{k} |\tilde{R}_{\ell,k}^\star|^2, \quad k = N_t - i + 1, \tag{3.4}$$

where $\tilde{\boldsymbol{h}}_k^\star$ is the kth column of reduced dual basis $\tilde{\boldsymbol{H}}^\star$. Then, the PEP in (3.3) is rewritten as

$$P(z_i \to \hat{z}_i | \tilde{\boldsymbol{H}}) = Q\left(\sqrt{\frac{2|e_i|^2}{\sigma_w^2 \sum_{\ell=1}^{k} |\tilde{R}_{\ell,k}^\star|^2}}\right), \quad k = N_t - i + 1. \tag{3.5}$$

Based on $P(\boldsymbol{s} \neq \hat{\boldsymbol{s}} | \boldsymbol{H}) \leq P(\boldsymbol{z} \neq \hat{\boldsymbol{z}} | \tilde{\boldsymbol{H}})$ [102] and the PEP results in (3.5), it is expected that the detector's overall error performance is better when the values of $\sum_{\ell=1}^{k} |\tilde{R}_{\ell,k}^\star|^2, 1 \leq k \leq N_t$, i.e., the norms of the column vectors of $\tilde{\boldsymbol{R}}^\star$, are smaller. Next, we will analyze how the execution of the LLL algorithms affect the norms of the column vectors in $\tilde{\boldsymbol{R}}^\star$.

To simplify the analysis, we provide an equivalent form of the LLL algorithm or LLL variant (fcLLL, greedy LLL, etc.) as shown in Table 3.1, which can also generates an LLL-reduced basis. In the table, a full size reduction (Lines 3-5) is performed before any LLL iterations. After the full size reduction, based on (2.20), we obtain that the off-diagonal entries of row k ($1 \leq k < N_t$) in $\tilde{\boldsymbol{R}}^\star$ are reduced as

$$|\tilde{R}_{k,n}^\star|^2 \leq \frac{1}{2} |\tilde{R}_{k,k}^\star|^2, \quad k < n \leq N_t. \tag{3.6}$$

Then, the loop of the LLL iterations are executed (Lines 6-10). Each time an LLL iteration with an appropriate index k is selected such that the relaxed Lovász condition (4.2) is not satisfied. Inside each LLL iteration, column swap (Line 7) is first performed followed by the full size reduction (Line 8). Note that full size reduction instead of effective size reduction is adopted here, so that the off-diagonal entries of each row in $\tilde{\boldsymbol{R}}^\star$ are reduced according to the diagonal entry in the same row as shown in (3.6) after each LLL iteration. The adopted modifications for size reduction in the equivalent form do not change the final error performance of the DLLL-aided LDs since the delayed full size reduction is always adopted in our greedy LLL algorithms.

Table 3.1: Equivalent Form of the LLL Algorithm and LLL Variants

Input: $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{P}$ (after QR/SQRD/MMSE-SQRD)
Output: $\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{R}}, \boldsymbol{T}$

1: Initialize: $\tilde{\boldsymbol{Q}} = \boldsymbol{Q}, \tilde{\boldsymbol{R}} = \boldsymbol{R}, \boldsymbol{T} = \boldsymbol{P}, \delta \in (1/2, 1], N_{max}$
2: $n_{iter} = 0$
3: **for** $k = 2 : N_t$
4: Execute ***size reduction*** (lines 4-10 of Table 2.1)
5: **end**
6: **for** select $k \in \{2, \ldots, N_t\}$ based on the LLL algorithm or LLL Variants
7: Execute ***column swap*** (Lines 12-15 of Table 2.1)
8: Execute ***full size reduction*** (Lines 3-5 of this table)
9: $n_{iter} = n_{iter} + 1$
10: **end**

Now we are ready to show how the selection of the LLL iterations affects the column norms of matrix $\tilde{\boldsymbol{R}}^{\star}$ in the equivalent form. Before the LLL iterations, the off-diagonal entries of each row have already be reduced according to the diagonal entry in the same row (Lines 3-5 of Table 3.1). During an LLL iteration (Lines 7-9 of Table 3.1), for the effect of the column swap, first, the operation of only swapping two columns (Line 12 of Table 2.1) does not affect the column norms of $\tilde{\boldsymbol{R}}^{\star}$; second, the operation of the matrix multiplication (Lines 13-14 of Table 2.1) does not affect the column norms of $\tilde{\boldsymbol{R}}^{\star}$ either, since $\boldsymbol{\Theta}$ is a unitary matrix. Therefore, the column swap (Line 7 of Table 3.1) does not change the column norms of $\tilde{\boldsymbol{R}}^{\star}$. However, the column swap does affect the values of the diagonal entries in $\tilde{\boldsymbol{R}}^{\star}$ as the following proposition states:

Proposition 3.1 *During an LLL iteration, when the column swap happens at column pair* $(k-1, k)$, $|\tilde{R}^{\star}_{k-1,k-1}|^2$ *decreases and* $|\tilde{R}^{\star}_{k,k}|^2$ *increases. Furthermore, the decreased value from* $|\tilde{R}^{\star}_{k-1,k-1}|^2$ *is greater than the increased value from* $|\tilde{R}^{\star}_{k,k}|^2$.

Proof See Appendix B.

Proposition 1 indicates that during the following full size reduction (Line 8 of Table 3.1), only the off-diagonal entries of the $(k-1)$th row in $\tilde{\boldsymbol{R}}^{\star}$ would be further

reduced because $|\tilde{R}^{\star}_{k-1,k-1}|^2$ is largely decreased. Since up to $N_t - k + 2$ off-diagonal entries can be reduced in the $(k-1)$th row, there can be up to $N_t - k + 2$ columns with decreased norms. To understand this procedure, an example is shown in Fig. 3.1 with $N_t = 4$. It can be seen that 2 column norms may be reduced if the LLL iteration is selected with $k = 3$, while 3 column norms may be reduced if the LLL iteration is selected with $k = 2$ (columns with reduced norm are shown in ellipses). It is obvious that a smaller k leads to more number of columns in $\tilde{\boldsymbol{R}}^{\star}$ with decreased norms, and better PEP of the overall $\boldsymbol{z}$ can be obtained based on (3.5). Thus, it is preferable to select the LLL iteration with smaller k each time to obtain better error performance in the DLLL-aided LDs.

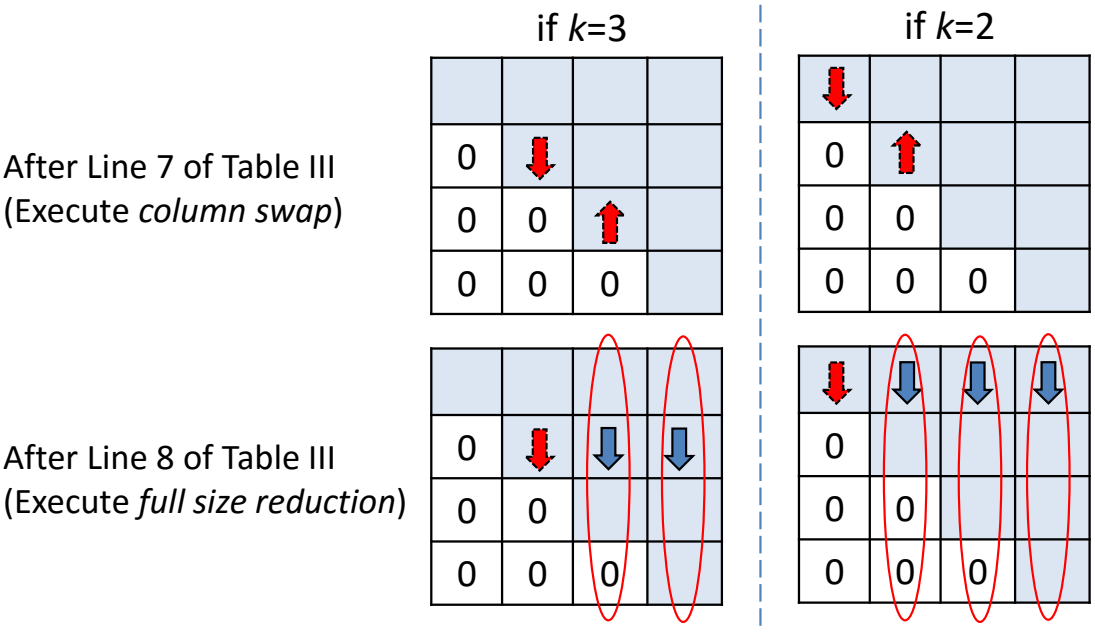


Figure 3.1: The change of matrix $\tilde{\boldsymbol{R}}^{\star}$ with different LLL iteration selection when $N_t = 4$. *Two* column norms may be reduced if selecting the LLL iteration with $k = 3$, while *three* column norms may be reduced if selecting the LLL iteration with $k = 2$.

3.2 Error Performance Analysis of LLL-aided SIC/K-best Detectors

Since we consider complex-valued detectors, the LLL-aided SIC detector is a special case of LLL-aided K-best detector where the number of candidates is fixed as one in the K-best detection. Therefore, we focus on the error performance analysis of LLL-aided SIC detectors to find the effect of the LLL algorithms. For the LLL-aided SIC detectors, the estimation is operated from the N_tth symbol to the 1st symbol. For simplicity, let us ignore the error propagation. Then, based on (2.17), the resulting layers can be modeled as

$$\hat{y}_k = \tilde{R}_{k,k} z_k + \tilde{n}_k, \quad 1 \leq k \leq N_t. \tag{3.7}$$

where $\hat{y}_k = \tilde{y}_k - \sum_{\ell=k+1}^{N_t} \tilde{R}_{k,\ell} z_\ell$. Thus, the error rate of the kth symbol that z_k is erroneously detected as $\hat{z}_k \neq z_k$ given the reduced channel $\tilde{\boldsymbol{H}}$, can be derived as

$$P\left(z_k \to \hat{z}_k | \tilde{\boldsymbol{H}}\right) = Q\left(\sqrt{\frac{2|e_k|^2 \tilde{R}_{k,k}^2}{\sigma_w^2}}\right), \tag{3.8}$$

which means that the larger $\tilde{R}_{k,k}^2$, the smaller the error rate of z_k.

When the LLL iteration is selected with column swap at the column pair $(k-1, k)$, the value of $\tilde{R}_{k,k}^2$ increases (see the proof of Proposition 3.1). Thus better error rate of z_k can be obtained based on (3.8). It is well known that the achievable performance of SIC/K-best detectors is limited by the error propagation, which means that the correctness of the first few detected layers is crucial to the overall error performance. Therefore, it is preferable to select the LLL iteration with larger value of k each time to obtain better error performance in the LLL-aided SIC/K-best detectors.

CHAPTER IV

ENHANCED GREEDY LLL ALGORITHMS

In this chapter, we propose two enhanced greedy LLL algorithms for LR-aided MIMO detectors. First, we design a relaxed Lovász condition for searching the candidate set of LLL iterations with column swap operations. This relaxation does not need size reduction operations so that it can save complexity compared to the existing greedy LLL algorithms. Then, we propose two alternatives to select the optimal one in the candidate set of LLL iterations. One is based on a relaxed criterion of the decrease in LLL potential, the other is based on the error performance of dual-LR-aided LDs and LR-aided SIC/K-best detectors as we discussed in the last chapter. Furthermore, we prove that the proposed two algorithms achieve full receive diversity in the dual-LR-aided LDs and LR-aided SIC detectors. Lastly, simulations show that the proposed two algorithms not only converge faster but also exhibit much lower complexity than the existing greedy LLL variants [97, 99] while the error performance is maintained. The contents of this chapter are based on our publications [39, 71, 74].

4.1 Finding the Candidate Set of LLL Iterations: Relaxing the Lovász Condition

To search the candidate set of LLL iterations with column swap operations, the PSLLL-OSSC algorithm [99] firstly applies size reduction to the elements $\tilde{\boldsymbol{R}}_{1:k-1,k}$ in each column pair $(k-1,k)$, and then checks the Lovász condition of all column pairs to find the candidate set of LLL iterations. The GDR algorithm [97] adopts the same procedure as that of the PSLLL-OSSC, except that the size reduction is only applied to one element $\tilde{R}_{k-1,k}$ in each column pair $(k-1,k)$. The size reduction is unavoidable when searching the candidate set in both PSLLL-OSSC and GDR,

since the size reduction may update the element $\tilde{R}_{k-1,k}$ used in the Lovász condition evaluation.

Different from the PSLLL-OSSC and GDR algorithms, our enhanced greedy LLL algorithms do not perform size reduction when searching the candidate set of LLL iterations (the size reduction only exists inside the LLL iterations after the searching stage). To achieve this, we relax the Lovász condition in (2.21) as

$$\delta \leq \frac{|\tilde{R}_{k,k}|^2}{|\tilde{R}_{k-1,k-1}|^2} + \frac{|\tilde{R}_{k-1,k}|^2}{|\tilde{R}_{k-1,k-1}|^2} \leq \frac{|\tilde{R}_{k,k}|^2}{|\tilde{R}_{k-1,k-1}|^2} + \frac{1}{2}, \tag{4.1}$$

where the second inequality comes from (2.20) by assuming the size reduction has been performed before checking the Lovász condition. To prevent the performance degradation with this relaxation, we fix $\delta = 1$. Then, we obtain the relaxed Lovász condition adopted in the enhanced greedy LLL algorithms as

$$\frac{|\tilde{R}_{k,k}|}{|\tilde{R}_{k-1,k-1}|} \geq \frac{1}{\sqrt{2}}, \quad \forall\ 2 \leq k \leq N_t. \tag{4.2}$$

Note that the condition in (4.2) is also a special case of the Siegel condition in [18]. Since the size reduction does not affect the diagonal elements in matrix $\tilde{\boldsymbol{R}}$, one can use the condition (4.2) to search the candidate set of LLL iterations without performing size reduction.

4.2 Selecting the LLL Iteration: Relaxing the Decrescence of LLL Potential or Improving the Error Performance under Limited Iterations

To select an appropriate LLL iteration in the candidate set of LLL iterations each time, one may take the convergence characteristic of the LLL algorithm into consideration, so that the adopted selection scheme makes the greedy LLL algorithms converge fast. As discussed in the previous chapter, the convergence of LLL algorithms can be characterized by LLL potential D. The size reduction does not change the value of D, since the diagonal elements in $\tilde{\boldsymbol{R}}$ are unchanged. The value of D only

changes after a column swap. It has been shown that D is monotonically decreasing during the execution of the LLL algorithm and there is a lower bound for D that depends on $\mathcal{L}(\boldsymbol{H})$ [29]. Hence, the LLL algorithm would terminate after a finite number of LLL iterations.

Based on the characteristic of D, it is straightforward to select the LLL iterations such that the decrescence of D is maximized each time in order to accelerate the LLL's termination. Suppose that an LLL iteration happens at index k. Then the diagonal elements $\tilde{R}_{k-1,k-1}$, $\tilde{R}_{k,k}$ and the LLL potential D are updated as $\tilde{R}'_{k-1,k-1}$, $\tilde{R}'_{k,k}$ and D_k after the column swap, respectively, where D_k is calculated as

$$D_k = \prod_{\substack{i=1 \\ i\neq k-1,k}}^{N_t-1} |\tilde{R}_{i,i}|^{2(N_t-i)} |\tilde{R}'_{k-1,k-1}|^{2(N_t-k+1)} |\tilde{R}'_{k,k}|^{2(N_t-k)} = \frac{|\tilde{R}_{k-1,k}|^2 + |\tilde{R}_{k,k}|^2}{|\tilde{R}_{k-1,k-1}|^2} D. \quad (4.3)$$

At each LLL iteration, we select the column index k such that the decrescence of D is maximized

$$k = \arg \max_{2\leq k\leq N_t} \Delta D_k, \quad \text{with } \Delta D_k \stackrel{\text{def}}{=} D - D_k, \quad (4.4)$$

where ΔD_k is calculated as

$$\Delta D_k = \left(1 - \frac{|\tilde{R}_{k-1,k}|^2 + |\tilde{R}_{k,k}|^2}{|\tilde{R}_{k-1,k-1}|^2}\right) D. \quad (4.5)$$

The formulation of (4.4) is equivalent to the selection method of LLL iterations used in PSLLL-OSSC [99] and GDR [97]. It can be seen that ΔD_k in (4.5) depends on $\tilde{R}_{k-1,k}$ that may be changed after size reduction, which indicates that the size reduction must be performed before the execution of (4.4).

To avoid size reduction operations, we relax the ΔD_k term by assuming that the size reduction has been performed. Then, by updating the $|\tilde{R}_{k-1,k}|^2$ of ΔD_k in (4.5) based on (2.20), we obtain

$$\Delta D_k \geq \left(1 - \frac{\frac{1}{2}|\tilde{R}_{k-1,k-1}|^2 + |\tilde{R}_{k,k}|^2}{|\tilde{R}_{k-1,k-1}|^2}\right) D \stackrel{\text{def}}{=} \Delta D'_k. \quad (4.6)$$

According to $\Delta D'_k$, the relaxed version of (4.4) is formulated as

$$k = \arg \max_{2 \le k \le N_t} \Delta D'_k = \arg \min_{2 \le k \le N_t} \frac{|\tilde{R}_{k,k}|}{|\tilde{R}_{k-1,k-1}|}. \tag{4.7}$$

It can be seen that the proposed selection scheme of the LLL iterations in (4.7) is less complex than that in (4.4). In addition, its main calculation part is the same as the aforementioned relaxed Lovász condition in (4.2), which is denoted as

$$\eta_k = \frac{|\tilde{R}_{k,k}|}{|\tilde{R}_{k-1,k-1}|}, \quad 2 \le k \le N_t. \tag{4.8}$$

Since η_k depends on the diagonal elements of $\tilde{\boldsymbol{R}}$ and the LLL iteration at index k affects the diagonal elements $\tilde{R}_{k,k}$ and $\tilde{R}_{k-1,k-1}$, we only need to update η_{k-1} (if $k > 2$), η_k, and η_{k+1} (if $k < N_t$) after the first LLL iteration instead of calculating all η_k each time.

To select an appropriate LLL iteration each time, another scheme is that we can take the error performance of different LR-aided detectors into consideration as discussed in Chapter 3, so that the adopted selection scheme can make the greedy LLL-aided MIMO detectors achieve near the same error performance as that of the original LLL-aided detectors with only a few LLL iterations, which is especially suitable for practical implementation. Specifically, we select the LLL iteration with smaller k each time to obtain better error performance in the DLLL-aided LDs, while select the LLL iteration with larger value of k each time to obtain better error performance in the LLL-aided SIC/K-best detectors.

4.3 Summary of the Two Enhanced Greedy LLL Algorithms

Based on the aforementioned two sections about finding the candidate set of LLL iterations and selecting the LLL iteration, we summarize the first greedy LLL algorithm in Table 4.1. In this table, Lines 2-5 are the initial evaluation of the proposed selection method of LLL iterations, and Lines 10-13 correspond to the updates of η_k and the next selection of LLL iteration. Line 7 is the condition to determine whether

the algorithm is terminated or not. In Line 7, besides the relaxed Lovász condition, the early termination is also adopted to set a predefined maximum iteration number as N_{max} ($N_{max} = \infty$ if there is no early termination). For the size reduction in Table 4.1, we adopt the effective size reduction (Line 8) as in ELLL plus the delayed full size reduction (DFSR, Lines 16-18) as in PSLLL-OSSC. By designing this structure, in the dual-LR-aided LDs, the proposed greedy LLL algorithm guarantees that the output basis satisfies the size reduction condition in (2.20) when the maximum number of the LLL iteration is limited; while in the LR-aided SIC/K-best detectors, the DFSR part can be dropped without affecting the error performance.

Table 4.1: Proposed Greedy LLL Algorithm-I

Input: $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{P}$ (after QR/SQRD/MMSE-SQRD)
Output: $\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{R}}, \boldsymbol{T}$

1: Initialize: $\tilde{\boldsymbol{Q}} = \boldsymbol{Q}, \tilde{\boldsymbol{R}} = \boldsymbol{R}, \boldsymbol{T} = \boldsymbol{P}, N_{max}$
2: **for** $i = 2 : N_t$
3: $\eta_i = |\tilde{R}_{i,i}|/|\tilde{R}_{i-1,i-1}|$
4: **end**
5: $k = \arg\min_{2 \le i \le N_t} \eta_i$
6: $n_{iter} = 0$
7: **while** $(\eta_k < 1/\sqrt{2})$ && $(n_{iter} < N_{max})$
8: Execute ***effective size reduction*** (Lines 4-10 of Table 2.1)
9: Execute ***column swap*** (Lines 12-15 of Table 2.1)
10: **for** $i = \{k, k \pm 1\} \cap \{2, \ldots, N_t\}$
11: $\eta_i = |\tilde{R}_{i,i}|/|\tilde{R}_{i-1,i-1}|$
12: **end**
13: $k = \arg\min_{2 \le i \le N_t} \eta_i$
14: $n_{iter} = n_{iter} + 1$
15: **end**
16: †**for** $k = 2 : N_t$
17: Execute ***size reduction*** (Lines 4-10 of Table 2.1)
18: **end**

†Lines 16-18 can be dropped for LR-aided SIC/K-best detectors.

Similarly, the second enhanced greedy LLL algorithm is summarized in Table 4.2. In the table, the relaxed Lovász condition (Lines 3 and 11) is expressed in an equivalent form as (4.2) but without division operation, which is more suitable for

high speed hardware implementation. Line 6 adopts the relaxed Lovász condition and maximum iteration number to determine whether the algorithm is terminated or not. For the selection of LLL iterations (Line 7), the min/max operation means to select the minimum/maximum index i, $i \in [2, N_t]$, such that the value of $flag(i)$ is one. This selection method is optimized corresponding to the error performance of specific LR-aided detectors as discussed above. For the size reduction in Table 4.2, we adopt the effective size reduction (Line 8) as in ELLL plus the delayed full size reduction (DFSR, Line 15) as in PSLLL-OSSC. By designing this structure, in the dual-LR-aided LDs, the enhanced greedy LLL algorithm guarantees that the output basis satisfies the size reduction condition in (2.20) when the maximum number of the LLL iteration is limited; while in the LR-aided SIC/K-best detectors, the DFSR part can be dropped without affecting the error performance.

Table 4.2: Proposed Greedy LLL Algorithm-II

Input: $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{P}$ (after QR/SQRD/MMSE-SQRD)
Output: $\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{R}}, \boldsymbol{T}$

1: Initialize: $\tilde{\boldsymbol{Q}} = \boldsymbol{Q}, \tilde{\boldsymbol{R}} = \boldsymbol{R}, \boldsymbol{T} = \boldsymbol{P}, flag = \text{zeros}(1, N_t), N_{max}$
2: **for** $i = 2 : N_t$
3: $\quad flag(i) = (\sqrt{2}|R_{i,i}| < |R_{i-1,i-1}|)$
4: **end**
5: $n_{iter} = 0$
6: **while** $(\text{sum}(flag) \neq 0)$ && $(n_{iter} < N_{max})$
7: $\quad k = \begin{cases} \arg\min_{2 \leq i \leq N_t, flag(i)==1}(i) & \text{if Dual-LR-aided LD} \\ \arg\max_{2 \leq i \leq N_t, flag(i)==1}(i) & \text{if LR-aided SIC} \end{cases}$
8: $\quad$ Execute ***effective size reduction*** (Lines 4-10 of Table 2.1)
9: $\quad$ Execute ***column swap*** (Lines 12-15 of Table 2.1)
10: $\quad$ **for** $i = \{k, k \pm 1\} \cap \{2, \ldots, N_t\}$
11: $\quad\quad flag(i) = (\sqrt{2}|R_{i,i}| < |R_{i-1,i-1}|)$
12: $\quad$ **end**
13: $\quad n_{iter} = n_{iter} + 1$
14: **end**
15: †Execute ***full size reduction*** (Lines 16-18 of Table 4.1)

†Line 15 can be dropped for LR-aided SIC/K-best detectors.

For MIMO detection, one common metric to measure performance is the diversity

order, which has the following definition:

Definition 4.1 (Diversity Order [64]): The diversity order G_d is defined as

$$G_d = \lim_{\text{SNR}\to\infty} -\frac{\log(P_e(\text{SNR}))}{\log(\text{SNR})}, \tag{4.9}$$

where $P_e(\text{SNR})$ is the average error probability as a function of SNR for a given MIMO system.

Although our two enhanced greedy LLL algorithms adopt some modifications compared to the LLL and existing greedy LLL algorithms to achieve high speed and low complexity, the error performance of the LR-aided MIMO detectors is maintained in terms of diversity order as the following two propositions state:

Proposition 4.1 *The dual-LR-aided LDs with the two enhanced greedy LLL algorithms achieve the full receive diversity order Nr as the ML detector.*

Proof See Appendix C.

Proposition 4.2 *The LR-aided SIC detectors with the two enhanced greedy LLL algorithms achieve the full receive diversity order Nr as the ML detector.*

Proof See Appendix D.

4.4 Numerical Results and Discussion

In this section, we validate the bit error rate (BER), convergence, and complexity of the different LR algorithms, i.e., the proposed greedy-LLL-I and greedy-LLL-II, the PSLLL-OSSC [99], the GDR [97], and the LLL/ELLL algorithms, in dual-LR-aided MMSE and LR-aided MMSE-SIC detectors through simulations. All MIMO detectors are based on complex-valued implementation, and MMSE-SQRD is adopted for better BER performance, which could also reduce the number of LLL iterations [84]. In the simulations, a quasi-static flat fading channel is adopted where the entries of the MIMO channel are generated as i.i.d. complex Gaussian variables with zero mean

and unit variance. For the BER comparisons of different LRs, we evaluate the BER versus energy per bit to noise density that is defined as $E_b/N_0 = N_r\sigma_s^2/(\sigma_w^2 \log_2 \mathcal{M})$, and over 5000 bit errors are produced for each BER result.

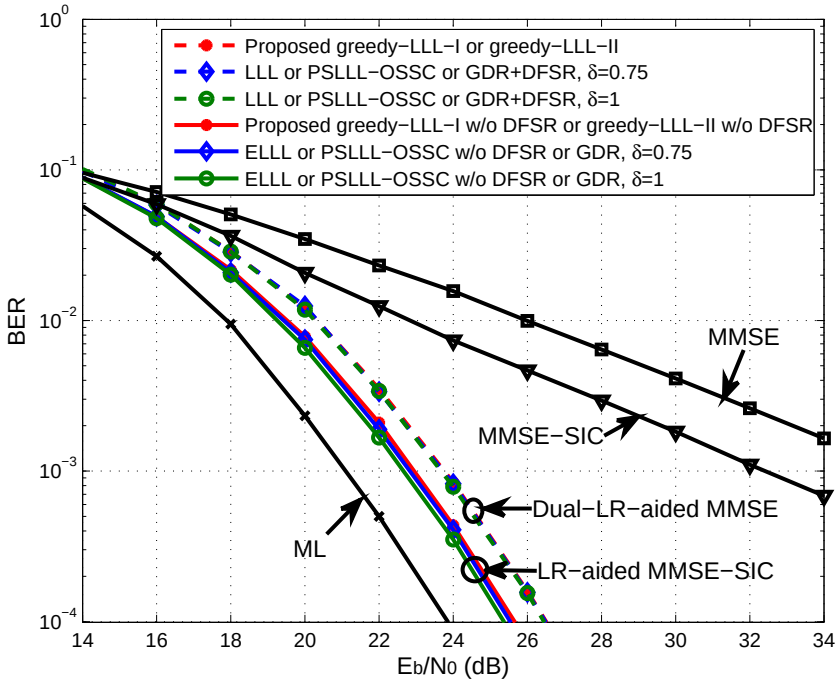
Note that if only the effective size reduction is adopted in the LR algorithm, the BER performance is unchanged in the LR-aided MMSE-SIC detector [18, 34], but this is not the case in the dual-LR-aided MMSE detector. To make fair comparison, we use the LRs with full size reduction in the dual-LR-aided MMSE detector, i.e., LLL, PSLLL-OSSC, GDR+DFSR (adding delayed full size reduction in the end of GDR), and the proposed greedy LLL-I/II; while we adopt the LRs with only effective size reduction in the LR-aided MMSE-SIC detector, i.e., ELLL, PSLLL-OSSC without DFSR, GDR, and the proposed greedy LLL-I/II without DFSR. For simplicity, we only use the aforementioned accurate description in the figures, while we do not explicitly distinct them in the context and just use LLL, PSLLL-OSSC, GDR, and the proposed greedy LLL-I/II to represent the corresponding versions.

4.4.1 BER Comparisons of Different LR-aided MIMO Detectors

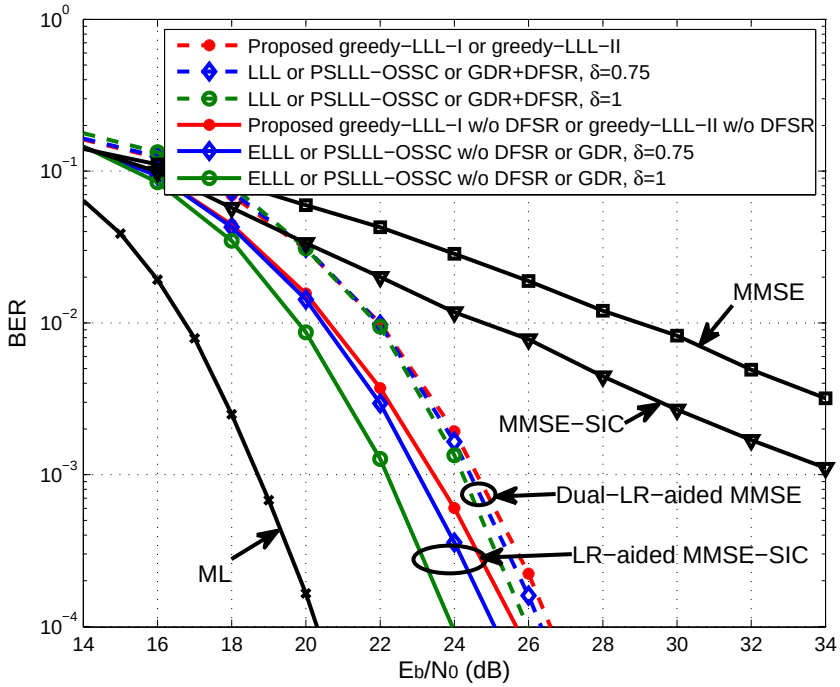
Fig. 4.1 depicts the uncoded BER performance of different LR algorithms in different LR-aided detectors for 4×4 and 8×8 MIMO systems with 64-QAM. As a comparison, the performances of MMSE, MMSE-SIC, and ML detectors are also provided.

First of all, the LLL, PSLLL-OSSC, and GDR algorithms have the same BER performance in different LR-aided MIMO detectors. Second, for each LR algorithm, the performance gain with $\delta = 1$ over $\delta = 0.75$ in the Lovász condition is marginal: for 4×4 MIMO systems, the gains are negligible; for 8×8 MIMO systems, the gains are only 1.1 dB and 0.3 dB at BER $= 10^{-4}$, in the LR-aided MMSE-SIC detector and dual-LR-aided MMSE detector, respectively. Considering the much slower convergence and higher complexity accompanied by $\delta = 1$ that will be shown later, $\delta = 0.75$

is a better complexity-performance tradeoff, which is also recommended in [29]. Finally, the proposed two greedy LLL algorithms enable the same BER performance for different LR-aided MIMO detectors and collect the full receive diversity order N_r as the ML detector does. Compared with the LLL/PSLLL-OSSC/GDR algorithm with $\delta = 0.75$, for 4×4 MIMO systems, the proposed two greedy LLL algorithms exhibit no performance loss in different LR-aided MIMO detectors; for 8×8 MIMO systems, the BER degradation of the proposed two greedy LLLs is only about 0.5 dB and 0.2 dB at BER = 10^{-4}, in LR-aided MMSE-SIC and dual-LR-aided MMSE detectors, respectively.



(a) BER of different LRs in a 4×4 MIMO system.



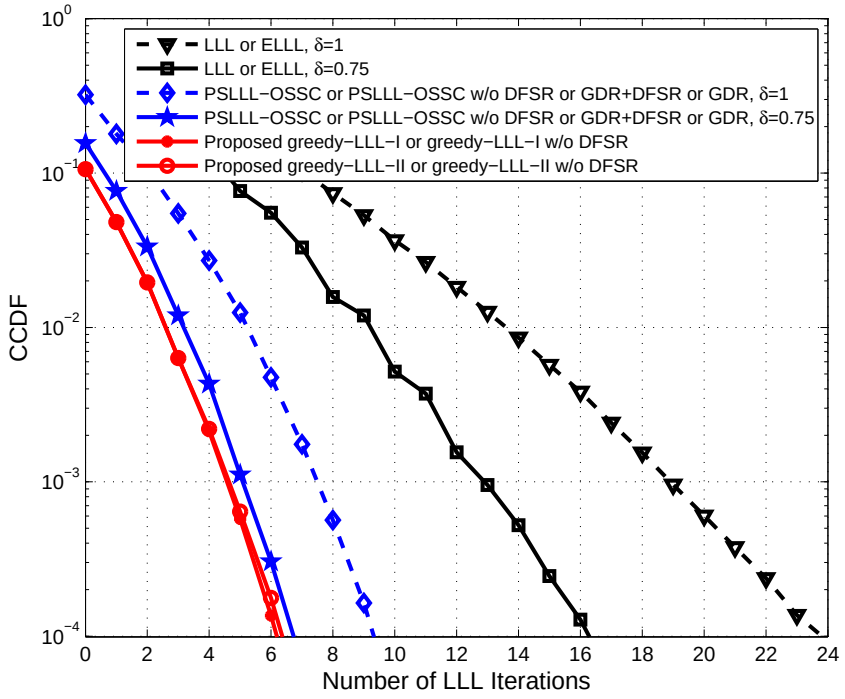
(b) BER of different LRs in an 8×8 MIMO system.

Figure 4.1: BER performance comparisons of dual-LR-aided MMSE and LR-aided MMSE-SIC detectors for 4×4 and 8×8 MIMO systems with 64-QAM.

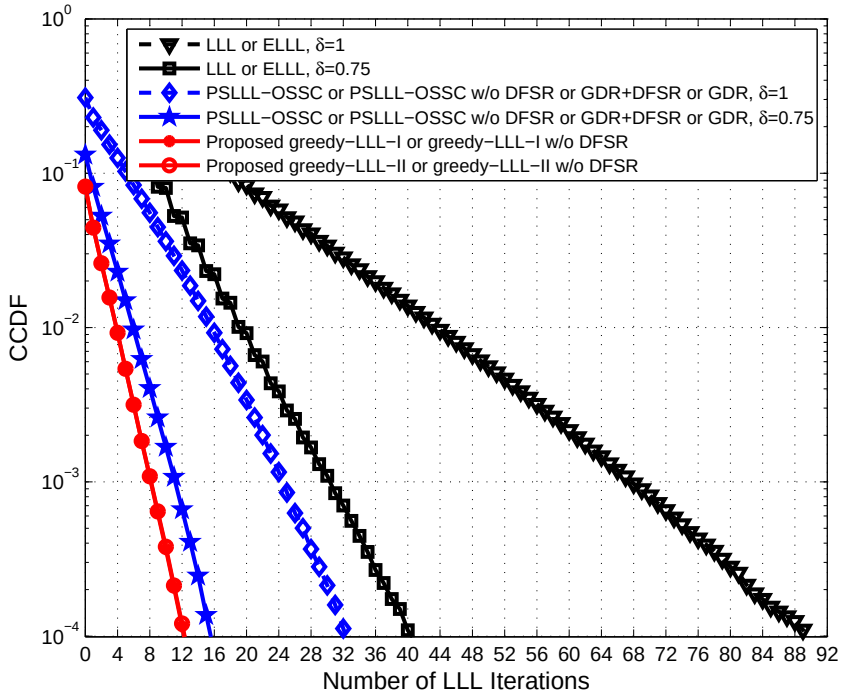
4.4.2 Convergence Comparisons of Different LR Algorithms

We compare the convergence of different LR algorithms in 4×4 and 8×8 MIMO systems, through the CCDF of the number of LLL iterations shown in Fig. 4.2, as well as the average and worst case numbers of LLL iterations summarised in Table 4.3. For each LR algorithm, 10^6 channel realizations are simulated, and the noise used in MMSE-SQRD is randomly generated such that the SNR is uniformly distributed from 0 dB to 40 dB (here we assume that σ_s^2 is normalized as $1/N_t$ so that SNR $= 1/\sigma_w^2$ per receiver antenna). Note that the convergence of each LR algorithm depends on the column swap only and size reduction does not affect it. Thus, the convergence performance with effective size reduction is the same as that with full size reduction in each LR algorithm.

First, the PSLLL-OSSC and GDR algorithms have the same performance since they adopt the same selection scheme of LLL iterations each time. Second, the LRs with $\delta = 1$ in the Lovász condition need more LLL iterations than those with $\delta = 0.75$ in LLL, PSLLL-OSSC, and GDR algorithms. Third, the PSLLL-OSSC/GDR converges much faster than the LLL due to the greedy characteristic, i.e., each LLL iteration comes along with a column swap. Finally, our proposed two greedy LLL algorithms have similar performance and can further improve the convergence compared to PSLLL-OSSC/GDR algorithms, especially in large MIMO dimensions. For example, compared to the PSLLL-OSSC/GDR algorithm with $\delta = 0.75$ in 8×8 MIMO systems as shown in Table 4.3, the proposed two algorithms save around 25% and 48% LLL iterations in the worst and average case, respectively.



(a) CCDF of LLL iterations in a 4×4 MIMO system.



(b) CCDF of LLL iterations in an 8×8 MIMO system.

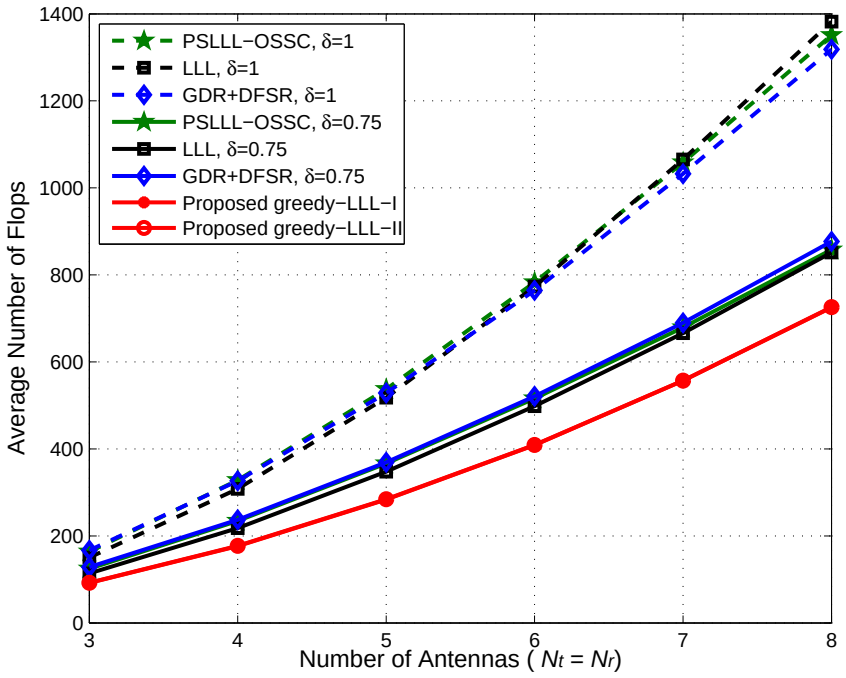
Figure 4.2: Convergence comparisons of different LR algorithms by CCDF of LLL iterations in 4×4 and 8×8 MIMO systems.

Table 4.3: The average and worst case numbers of LLL iterations in 4×4 and 8×8 MIMO systems.

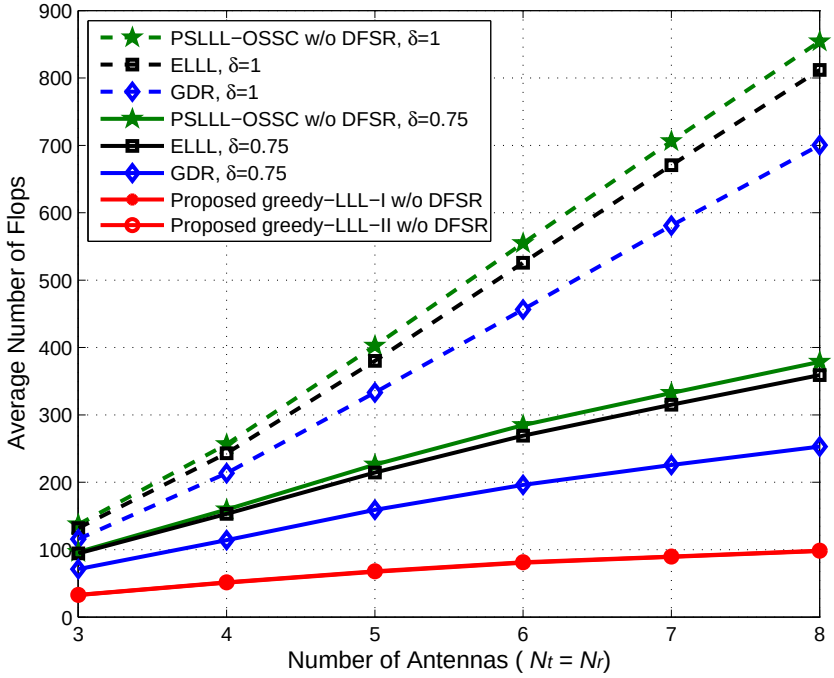
LR algorithm		4×4 MIMO		8×8 MIMO	
		average	worst	average	worst
LLL or ELLL	$\delta = 1$	4.2651	33	10.2483	125
	$\delta = 0.75$	3.4951	24	7.7278	62
GDR or PSLLL-OSSC	$\delta = 1$	0.7097	15	1.5361	48
	$\delta = 0.75$	0.2836	10	0.3659	28
Proposed algorithm-I		0.1825	10	0.1887	21
Proposed algorithm-II		0.1828	10	0.1889	21

4.4.3 Complexity Comparisons of Different LR Algorithms

To approximately evaluate the computational complexity of different LR algorithms, we compute the average equivalent real floating-point operations (flops) of different LRs from 3×3 to 8×8 MIMO systems in Fig. 4.3. For each LR algorithm, 10^5 channel realizations are simulated, and the method to generate noise in the MMSE-SQRD is the same as that in Section 4.4.2. The flops of each arithmetic operation are counted as follows: one flop for real operations like addition, subtraction, multiplication, division, comparison, absolute value, and square root; six flops for a complex multiplication; two flops for a multiplication between a real number and a complex number; two flops for a complex number divided by a real number; and two flops for rounding a complex number.



(a) Different LRs in dual-LR-aided MMSE detectors.



(b) Different LRs in LR-aided MMSE-SIC detectors.

Figure 4.3: Complexity comparisons of different LR algorithms from 3×3 to 8×8 MIMO systems.

First, for each LR scheme of LLL, PSLLL-OSSC, and GDR, the version with $\delta = 1$ in the Lovász condition exhibits higher complexity than the version with $\delta = 0.75$, and the complexity difference can even be doubled when only effective size reduction is adopted as shown in Fig. 4.3(b). This is because these LRs with $\delta = 1$ need more LLL iterations than those with $\delta = 0.75$ as discussed before. Second, the complexity difference is small among LLL, PSLLL-OSSC, and GDR under each δ value when full size reduction is adopted as shown in Fig. 4.3(a). However, the PSLLL-OSSC has higher complexity than the GDR, and even a little bit higher than the LLL under each δ value when only effective size reduction is adopted as shown in Fig. 4.3(b). The reason is that during the search for the candidate set of LLL iterations, the PSLLL-OSSC applies size reduction to the elements $\tilde{\boldsymbol{R}}_{1:k-1,k}$ in each column pair $(k-1, k)$, while GDR only applies size reduction to one element $\tilde{R}_{k-1,k}$ in each column pair $(k-1, k)$. Finally, our proposed two greedy LLL algorithms have almost the same complexity and achieve the lowest complexity thanks to the proposed relaxations and the faster convergence as demonstrated before. For example, even compared to the GDR algorithm with $\delta = 0.75$ when only effective size reduction is adopted as shown in Fig. 4.3(b), the proposed two algorithms save around 55% and 62% complexity in average for 4×4 and 8×8 MIMO systems, respectively.

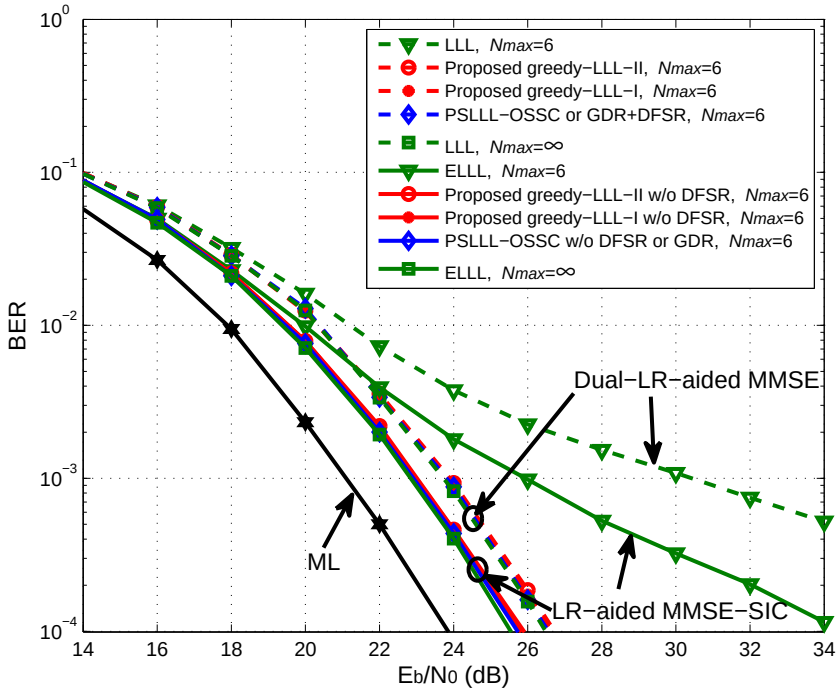
4.4.4 BER, Convergence, and Complexity of Different LR Algorithms with Early Termination

In this subsection, we consider the BER, convergence, and complexity of different LR algorithms with early termination, i.e., $N_{max} \neq \infty$, which is usually the case in hardware [18, 55]. Without loss of generality in the following simulations, we evaluate the 4×4 MIMO systems with 64-QAM, and adopt $\delta = 0.75$ in the PSLLL-OSSC, GDR, and LLL algorithms.

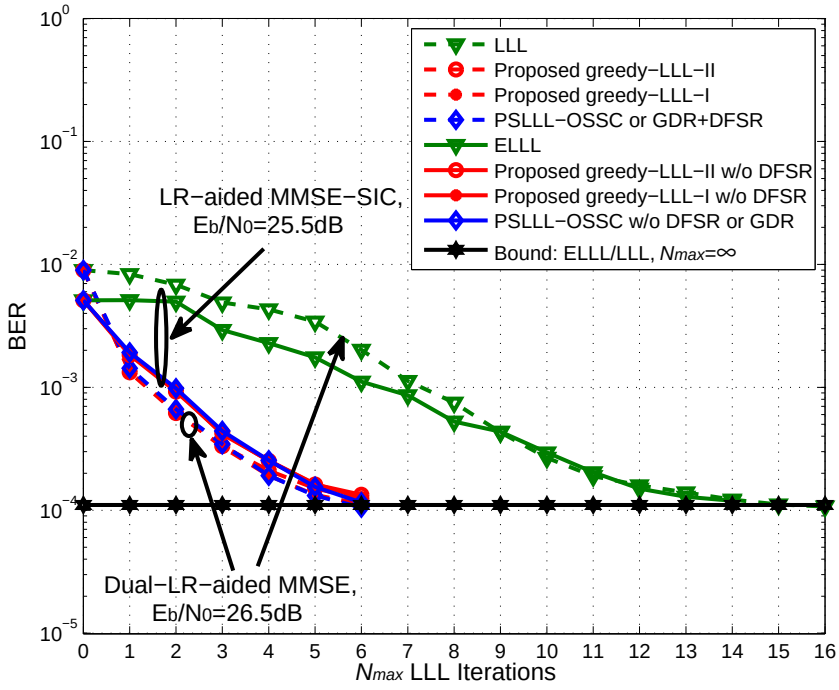
Fig. 4.4(a) depicts the results of BER versus E_b/N_0 of different LR algorithms in the dual-LR-aided MMSE detector and the LR-aided MMSE-SIC detector. Here

N_{max} is the minimum number of LLL iterations selected through simulations such that the PSLLL-OSSC/GDR with this N_{max} achieves similar error performance as the LLL without early termination. First, for PSLLL-OSSC/GDR in both the dual-LR-aided MMSE detector and the LR-aided MMSE-SIC detector, only $N_{max} = 6$ is needed to approach the best performance. Second, for the LLL algorithm with the same N_{max} as the PSLLL-OSSC/GDR, it has around 16dB and 8dB performance loss at BER= 10^{-4} in the dual-LR-aided MMSE detector and the LR-aided MMSE-SIC detector, respectively. Third, for both proposed greedy LLL algorithms with the same N_{max} as the PSLLL-OSSC/GDR, they achieve similar error performance as the LLL without early termination.

Fig. 4.4(b) demonstrates the convergence of different LR algorithms in terms of BER versus maximum number of LLL iteration N_{max}, where the E_b/N_0 is selected based on Fig. 4.4(a) such that the BER of the LLL with $N_{max} = \infty$ (without early termination) can achieve around 10^{-4}. Note that $N_{max} = 0$ denotes the corresponding degraded MIMO detectors without LR (i.e., from dual-LR-aided MMSE and LR-aided MMSE-SIC detectors to MMSE and MMSE-SIC detectors, respectively). First, the LLL algorithm exhibits much slower convergence speed than all greedy LLL algorithms in all cases. Second, the proposed two greedy LLL algorithms have almost the same convergence as the PSLLL-OSSC/GDR in both dual-LR-aided MMSE detector and the LR-aided MMSE-SIC detector, and all the greedy LLLs need only $N_{max} = 6$ LLL iterations to approach the best performance while the LLL algorithm needs $N_{max} = 14$ LLL iterations.



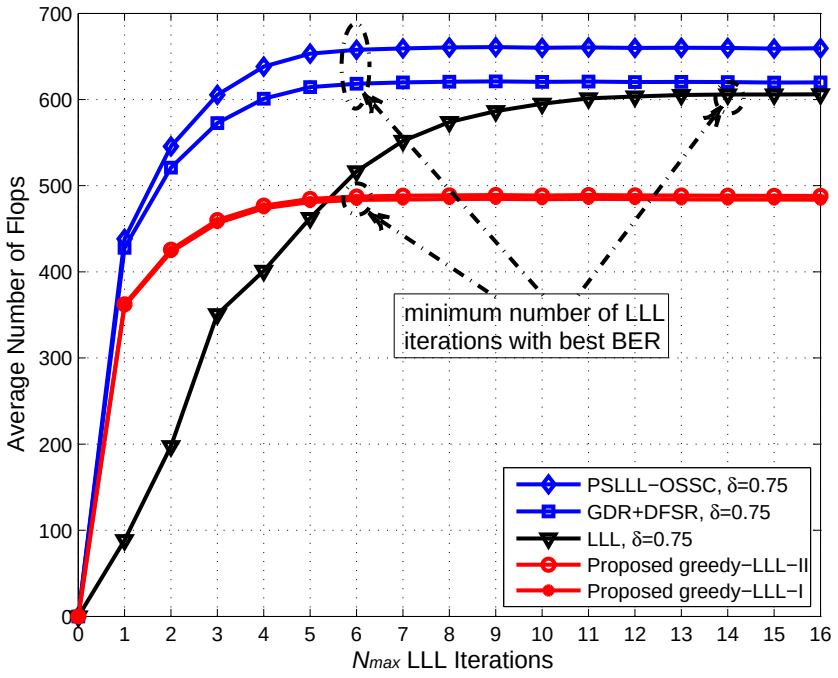
(a) Performance of BER versus E_b/N_0.



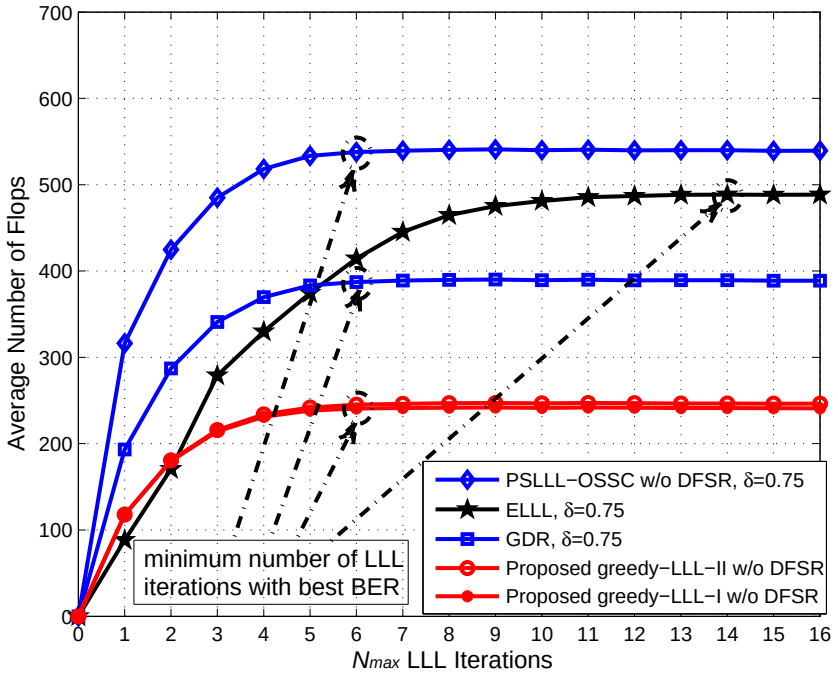
(b) Performance of BER versus number of LLL iterations.

Figure 4.4: BER comparisons of different LR algorithms with early termination in a 4×4 MIMO system with 64-QAM.

Fig. 4.5 shows the complexity of different LR algorithms in terms of average number of flops versus maximum number of LLL iteration N_{max}, where the values of E_b/N_0 are selected the same as those in Fig. 4.4(b). First, the complexity of each LR remains stable after some N_{max} LLL iterations. Based on the final stable complexities, the proposed two greedy LLL algorithms enable the lowest complexity. Second, when considering the minimum number of LLL iterations needed for the best achievable BER performance according to Fig. 4.4(b), the proposed two greedy LLL algorithms still exhibit the lowest complexity as shown in Fig. 4.5.



(a) Dual-LR-aided MMSE detectors with E_b/N_0=26.5dB.



(b) LR-aided MMSE-SIC detectors with E_b/N_0=25.5dB.

Figure 4.5: Complexity versus maximum number of LLL iterations of different LR algorithms with early termination in a 4×4 MIMO system with 64-QAM.

CHAPTER V

INCREMENTAL FIXED-COMPLEXITY LLL ALGORITHMS

In this chapter, we propose Incremental fixed-complexity LLL (Incremental fcLLL) algorithms for LR-aided MIMO detectors. The aim of the fcLLL algorithms is to solve the variable iteration order and complexity of the original LLL algorithm, such that the hardware implementation can be easier and more efficient. The existing fcLLL algorithms adopt fixed-column traverse strategy to process each column with equal priority, which is not optimized in terms of error performance and complexity. We propose enhanced fcLLL algorithms with novel column traverse strategies by allocating priorities to columns based on the characteristics of LLL and MIMO detection. In addition, we propose an improved termination criterion without sacrificing the error performance in the proposed fcLLL algorithms with maximum number of iterations. Simulations show that our proposed fcLLL algorithms converge faster than LLL and existing fcLLL algorithms, and yield better error performance than the LLL and existing fcLLL algorithms when the maximum number of LLL iterations is fixed. Furthermore, in large MIMO systems, our proposed fcLLL algorithms exhibit significant complexity advantage, saving about 90% LLL iterations in average compared to the existing fcLLL algorithms for a 128×128 MIMO system with 64-QAM. The contents of this chapter are based on our publications [72, 73, 75].

5.1 Incremental Column Traverse Strategies

The main principle in our algorithms is to allocate priorities for column index when performing column traverse among LLL iterations, by exploiting the relationship

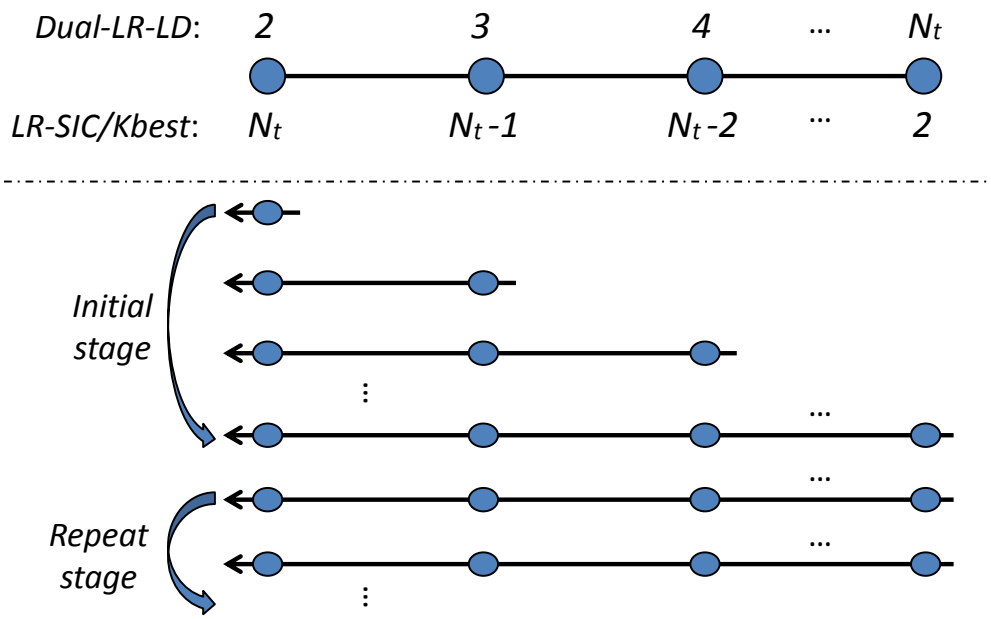


Figure 5.1: Column index sequence in the Incremental Sequential fcLLL algorithm using incremental sequential strategy.

between the LLL algorithms and the error performance of LR-aided MIMO detectors as discussed in Chapter 3. Specifically, for dual-LR-aided LDs, we start LLL iterations from $k = 2$ and use as many as possible $k = 2$ in the fcLLL algorithms; while in the LR-aided SIC/K-best detectors, we start LLL iterations from $k = N_t$ and use as many as possible $k = N_t$ in the fcLLL algorithms. To achieve this principle, we propose two improved column traverse strategies, i.e., incremental sequential strategy and incremental even-odd strategy, which are elaborated in the following.

For the incremental sequential strategy, the generation of the column index sequence is illustrated in Fig. 5.1. There the initial stage contains incremental subsequences with sequential index and increased length until the last sub-sequence contains all elements in $[2, N_t]$, and the repeat stage replicates the last sub-sequence in the initial stage. When the fcLLL is used in dual-LR-aided LDs, the column index sequence k starts from 2 and consists of

$$k = \underbrace{\mathbf{2}, 3, 2, \mathbf{4}, \mathbf{3}, \mathbf{2}, ..., N_t, N_t-1, ..., 2}_{\textbf{Initial stage}}, \underbrace{repeat[N_t, N_t-1, ..., 2]}_{\textbf{Repeat stage}}. \tag{5.1}$$

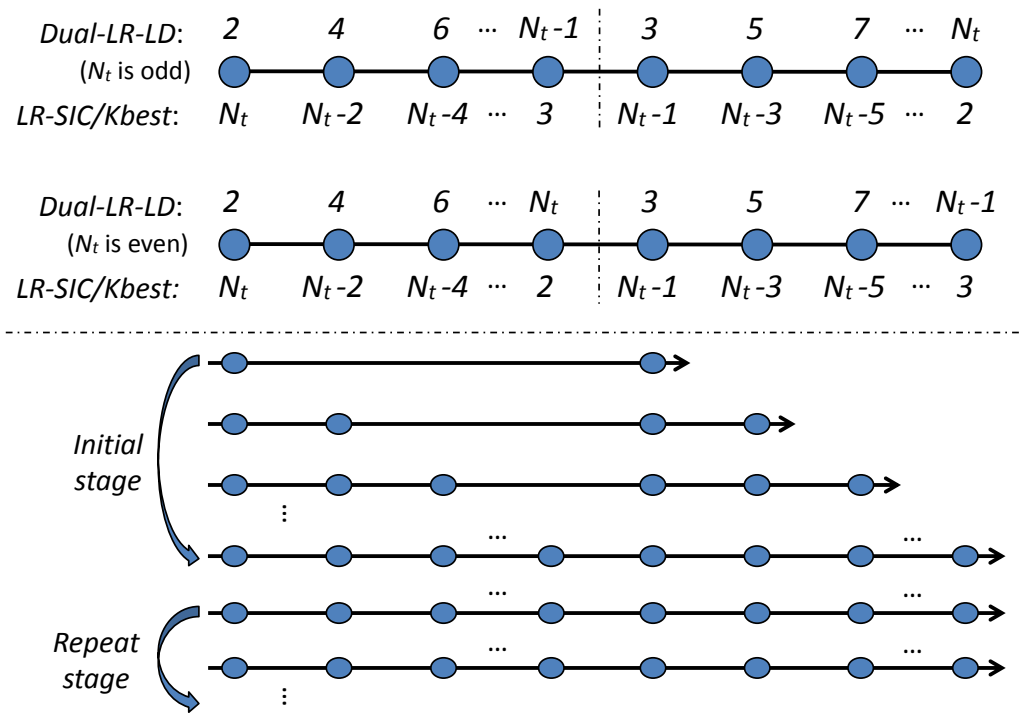


Figure 5.2: Column index sequence in the Incremental Even-odd fcLLL algorithm using incremental even-odd strategy.

Here the sub-sequences are highlighted by bold and italic letters alternatively. When the fcLLL is used in LR-aided SIC/K-best detectors, the sequence k starts from N_t and consists of

$$k = \underbrace{\mathbf{N_t}, N_t-1, N_t, \mathbf{N_t-2}, \mathbf{N_t-1}, \mathbf{N_t}, ..., 2, 3, ..., N_t}_{\textbf{Initial stage}}, \underbrace{repeat[2, 3, ..., N_t]}_{\textbf{Repeat stage}}. \tag{5.2}$$

For the incremental even-odd strategy, the generation of the column index sequence is illustrated in Fig. 5.2. There the initial stage contains incremental sub-sequences with even-odd index and increased length until the last sub-sequence contains all elements in $[2, N_t]$, and the repeat stage replicates the last sub-sequence in the initial stage. Suppose N_t is an even number (if N_t is odd, similar result can be obtained based on Fig. 5.2), when the fcLLL is used in dual-LR-aided LDs, the

column index sequence k starts from 2 and consists of

$$k = \underbrace{\mathbf{2}, \mathbf{3}, 2, 4, 3, 5, ..., \mathbf{2}, \mathbf{4}, ..., \mathbf{N_t}, \mathbf{3}, \mathbf{5}, ..., \mathbf{N_t - 1}}_{\textbf{Initial stage}}, \underbrace{repeat[\mathbf{2}, \mathbf{4}, ..., \mathbf{N_t}, \mathbf{3}, \mathbf{5}, ..., \mathbf{N_t - 1}]}_{\textbf{Repeat stage}}. \tag{5.3}$$

When the fcLLL algorithm is used in LR-aided SIC/K-best detectors, the column index sequence k starts from N_t and consists of

$$k = \underbrace{\mathbf{N_t}, \mathbf{N_t - 1}, N_t, N_t - 2, N_t - 1, N_t - 3, ..., \mathbf{N_t}, \mathbf{N_t - 2}, ...\mathbf{2}, \mathbf{N_t - 1}, \mathbf{N_t - 3}...\mathbf{3}}_{\textbf{Initial stage}}, \underbrace{repeat[\mathbf{N_t}, \mathbf{N_t - 2}, ...\mathbf{2}, \mathbf{N_t - 1}, \mathbf{N_t - 3}...\mathbf{3}]}_{\textbf{Repeat stage}}. \tag{5.4}$$

Note that the purpose of initial stage is to obtain fast performance improvement. Specifically, in the dual-LR-aided LDs, the idea is to start LLL iterations from $k = 2$ and use as many as possible $k = 2$ in the fcLLL algorithms; while in the LR-aided SIC/K-best detectors, the idea is to start LLL iterations from $k = N_t$ and use as many as possible $k = N_t$ in the fcLLL algorithms. Once the sub-sequence in initial stage contains all column index, the repeat stage starts such that each column index happens equally, which is also useful to obtain fast termination if we aim to obtain an LLL-reduced basis.

Our designed Incremental fcLLL algorithm with fixed number of iterations is summarized in Table 5.1, where one LLL iteration corresponds to Lines 3-8, N_{max} denotes the fixed number of LLL iterations, and the $kSeq$ of Line 1 corresponds to the sequence of column index k generated by the incremental sequential strategy or incremental even-odd strategy. We denote the designed enhanced fcLLL algorithm as Incremental Sequential fcLLL algorithm when using the incremental sequential column traverse strategy, and Incremental Even-odd fcLLL algorithm when using the incremental Even-odd strategy.

5.2 Improved Termination Criterion

Another form of fcLLL algorithms is to set maximum number of iterations instead of fixed number of iterations. And the fcLLL algorithm can terminate before achieving

Table 5.1: The Incremental fcLLL Algorithm with Fixed Number of Iterations

Input: $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{P}$ (after QR/SQRD/MMSE-SQRD)
Output: $\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{R}}, \boldsymbol{T}$

1: Initialize: $\tilde{\boldsymbol{Q}} = \boldsymbol{Q}, \tilde{\boldsymbol{R}} = \boldsymbol{R}, \boldsymbol{T} = \boldsymbol{P}, \delta \in (1/2, 1], N_{max}, kSeq^{\dagger}$
2: **for** $n_{iter} = 1 : 1 : N_{max}$
3: $k = kSeq(n_{iter})$
4: Execute ***(effective) size reduction*** (lines 4-10 of Table 2.1)
5: **if** $\delta|\tilde{R}_{k-1,k-1}|^2 > |\tilde{R}_{k,k}|^2 + |\tilde{R}_{k-1,k}|^2$
6: Execute ***column swap*** (lines 12-15 of Table 2.1)
7: **end**
8: $n_{iter} = n_{iter} + 1$
9: **end**
10: ‡**for** $k = 2 : N_t$
11: Execute ***size reduction*** (lines 4-10 of Table 2.1)
12: **end**

$^{\dagger}kSeq$ can be based on either the incremental sequential strategy or the incremental even-odd strategy.
‡Lines 16-18 can be dropped for LR-aided SIC/K-best detectors.

the maximum number of iterations if an LLL-reduced basis is obtained. In this case, the existing fcLLLs terminate when there is no *column swap* in a super-iteration. Actually we can save some LLL iterations by exploiting the fixed structure of the column traverse in fcLLL algorithms. To do it, we define flag $CSflag$ with $(N_t + 1)$ bits to track the *column swap* operation for each index k as

$$CSflag(k) \stackrel{\text{def}}{=} \begin{cases} 0 & \textit{column swap} \text{ will not happen for index } k \\ 1 & \textit{column swap} \text{ may happen for index } k \end{cases}$$

where $k = 1, 2, \ldots, N_t + 1$. The $CSflag(1)$ and $CSflag(N_t + 1)$ are nil bits used to simplify the operation of Line 11 in Table 5.2 so that we do not need to consider whether k is boundary value like 2 or N_t. The $CSflag$ is initialized to ones and updated as

$$\begin{cases} CSflag(k) = 0 & \text{when no } \textit{column swap} \text{ at index } k \\ CSflag(k-1:1:k+1) = 1 & \text{when } \textit{column swap} \text{ at index } k \end{cases}$$

The *column swap* at k will lead to possible *column swap* at $k \pm 1$, which means $CSflag(k \pm 1) = 1$. Meanwhile, since *column swap* at k decreases $\tilde{R}_{k-1,k-1}$ and

increases $\tilde{R}_{k-1,k}$, it is possible that there is *size reduction* for $\tilde{R}_{k-1,k}$, which can result in *column swap* at k next time. Therefore, $CSflag(k)$ is set as 1 as well if *column swap* happens at k. As soon as the middle $N_t - 1$ indices are zeros in $CSflag$, which means there will be no *column swap* any more, the proposed fcLLL terminates with an LLL-reduced basis.

Our designed Incremental fcLLL algorithms with maximum number of iterations is summarized in Table 5.2, where one LLL iteration corresponds to lines 6-14, N_{max} denotes the maximum number of LLL iterations, the $kSeq$ of Line 1 corresponds to the sequence of column index k generated by the incremental sequential strategy or incremental even-odd strategy, and the $CSflag$ is used for tracking column swap operations.

Table 5.2: The Incremental fcLLL Algorithm with Maximum Number of Iterations

Input: $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{P}$ (after QR/SQRD/MMSE-SQRD)
Output: $\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{R}}, \boldsymbol{T}$

1: Initialize: $\tilde{\boldsymbol{Q}} = \boldsymbol{Q}, \tilde{\boldsymbol{R}} = \boldsymbol{R}, \boldsymbol{T} = \boldsymbol{P}, \delta \in (1/2, 1], N_{max}, kSeq^{\dagger}$
2: $CSflag = ones(1, N_t + 1)$
3: $n_{iter} = 1$
4: $kSeq_{idx} = 1$
5: **while** $(n_{iter} \leq N_{max})$ && $(sum(CSflag(2:1:N_t)) \neq 0)$
6: $k = kSeq(kSeq_{idx})$
7: $CSflag(k) = 0$
8: Execute ***(effective) size reduction*** (lines 4-10 of Table 2.1)
9: **if** $\delta|\tilde{R}_{k-1,k-1}|^2 > |\tilde{R}_{k,k}|^2 + |\tilde{R}_{k-1,k}|^2$
10: Execute ***column swap*** (lines 12-15 of Table 2.1)
11: $CSflag(k-1:1:k+1) = 1$
12: **end**
13: $n_{iter} = n_{iter} + 1$
14: $kSeq_{idx} = kSeq_{idx} + 1$
15: **end**
16: ‡**for** $k = 2 : N_t$
17: Execute ***size reduction*** (lines 4-10 of Table 2.1)
18: **end**

†$kSeq$ can be based on either the incremental sequential strategy or the incremental even-odd strategy.

‡Lines 16-18 can be dropped for LR-aided SIC/K-best detectors.

5.3 Numerical Results

In this section, the complexity and BER of the different LR algorithms, i.e., the proposed two Incremental fcLLLs, the Sequential fcLLL, the Even-odd fcLLL, and the LLL, are compared in dual-LR-aided LDs and LR-aided SIC detectors. In all LR algorithms, the parameter $\delta = 3/4$ is adopted for performance-complexity tradeoff as in [29]. The channels are frequency-flat and quasi-static fading whose entries are i.i.d. complex Gaussian variables with zero mean and unit variance. The SNR is given by $E_b/N_0 = N_r\sigma_s^2/(\sigma_w^2 log_2|\mathcal{M}|)$. For each BER value, a sufficient channel realizations are simulated so that over 3000 bit errors are generated.

5.3.1 Complexity Comparisons of Different LR Algorithms

The complexity are evaluated by the number of LLL iterations, since the major complexity components of each LLL iteration in different LR algorithms are the same, i.e., the size reduction, the Lovász condition, and the column swap. Also, in practical high-throughput pipelining implementation of the LLL algorithm, the complexity and latency are roughly linearly proportional to the number of LLL iterations [55]. We illustrate the LLL iterations of different LR algorithms in different antenna configurations by setting $N_{max} = \infty$ as in Table 5.2, i.e., no early termination is applied and LLL-reduced basis is guaranteed, so that all the LR algorithms would have the same BER performance in the LR-aided MIMO detectors. For each LR algorithm, 10^6 channel realizations are simulated.

Fig. 5.3 shows the complementary cumulative distribution function (CCDF) of the LLL iterations for different LR algorithms in 4×4 and 8×8 MIMO systems. It can be seen that the number of LLL iterations increases with the dimension of the MIMO systems. In both MIMO cases, our proposed two fcLLLs achieve the best performance among all the LR algorithms. Fig. 5.4 demonstrates the average value and standard deviation of the LLL iterations of different LR algorithms from

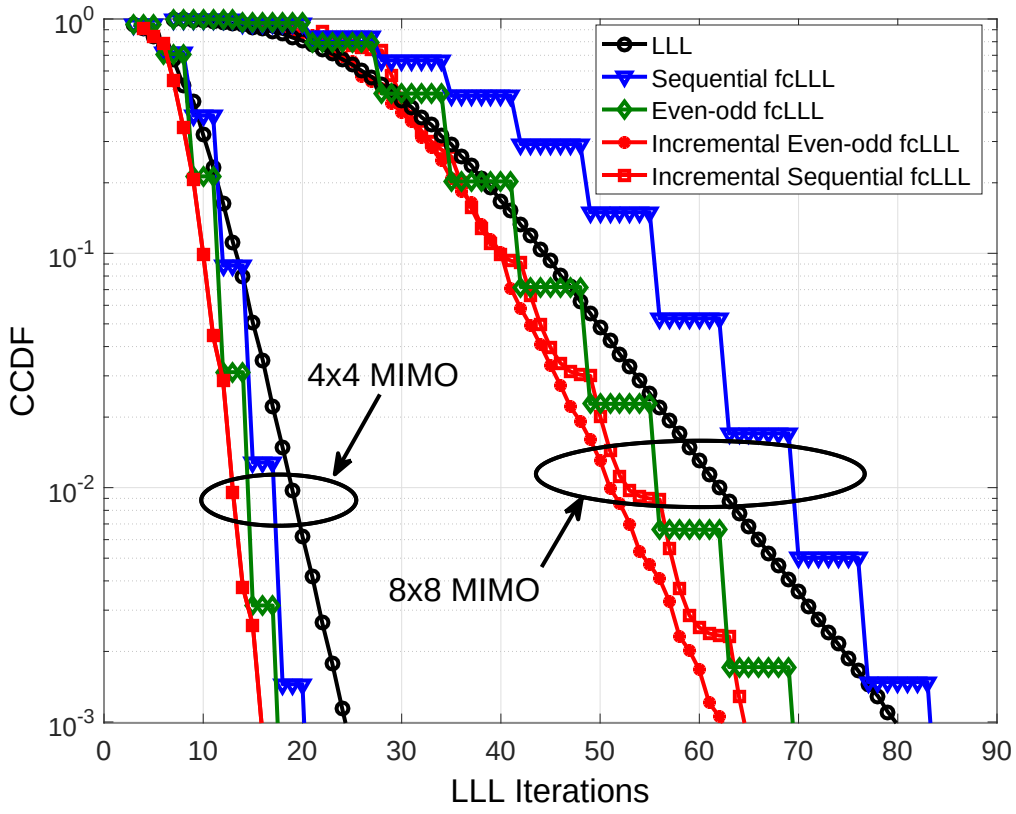


Figure 5.3: CCDFs of LLL iterations of different LR algorithms for 4×4 and 8×8 MIMO systems.

3×3 to 8×8 MIMO systems. First, the Even-odd fcLLL has similar average value compared with the LLL algorithm except $N_t = 8$ (a little worse than the LLL), but Even-odd fcLLL exhibits better performance in standard deviation. Second, among all the LR algorithms, the Sequential fcLLL has the worst average value as well as almost the worst standard deviation. However, our proposed two fcLLLs achieve the best complexity performance among all the LR algorithms in both average value and standard deviation. For example of the 8×8 MIMO case, our proposed two fcLLLs could save 8% and 23% LLL iterations in average compared to the Even-odd fcLLL and the Sequential fcLLL, respectively.

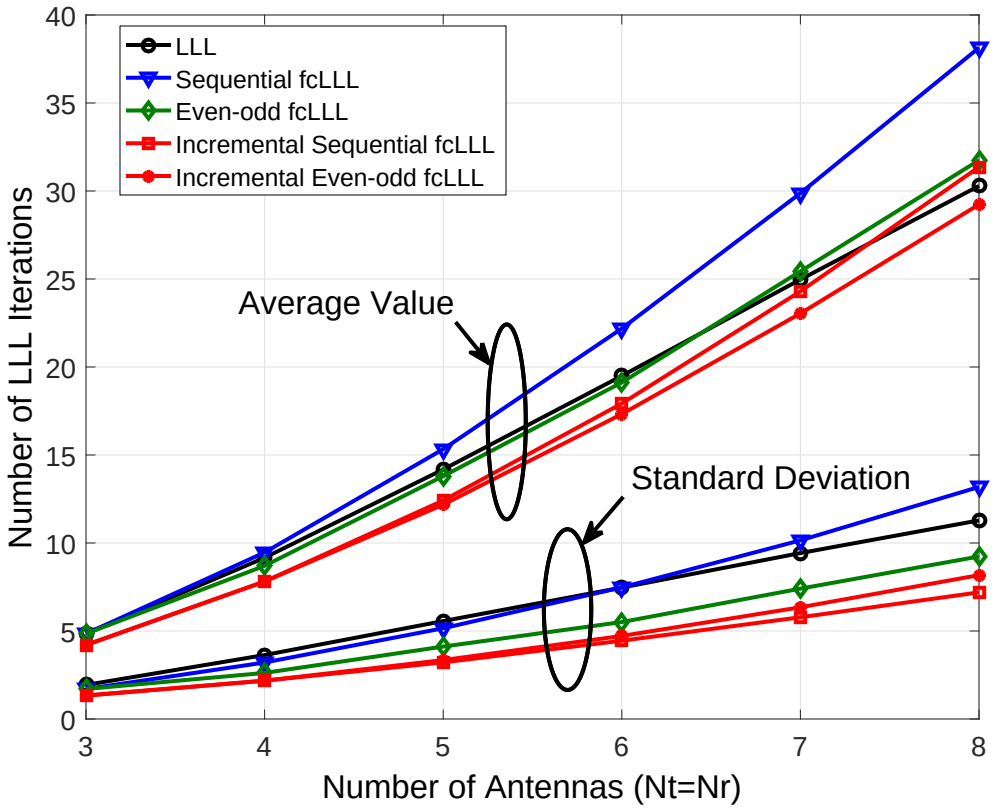
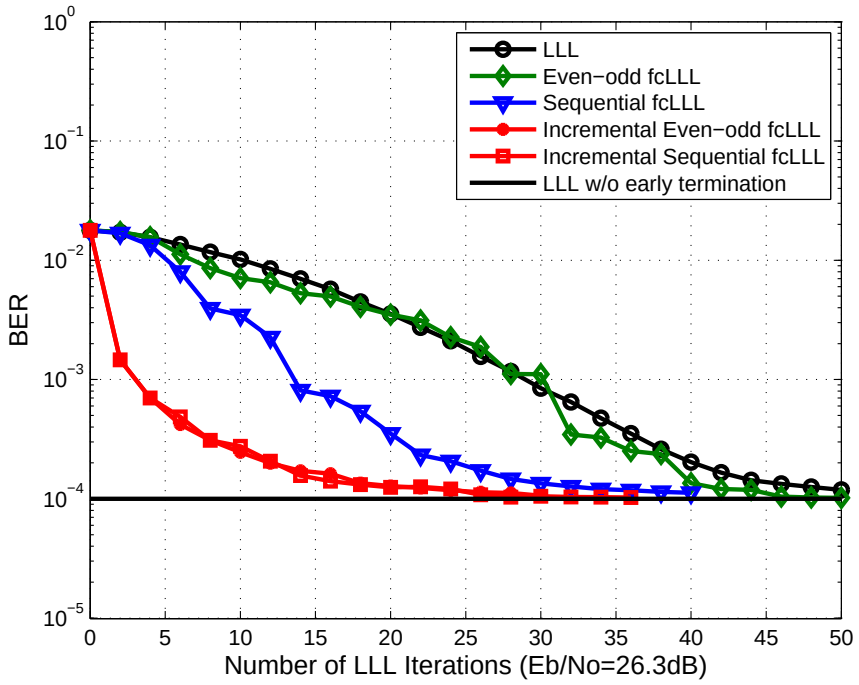


Figure 5.4: Average value and standard deviation of LLL iterations of different LR algorithms from 3×3 to 8×8 MIMO systems.

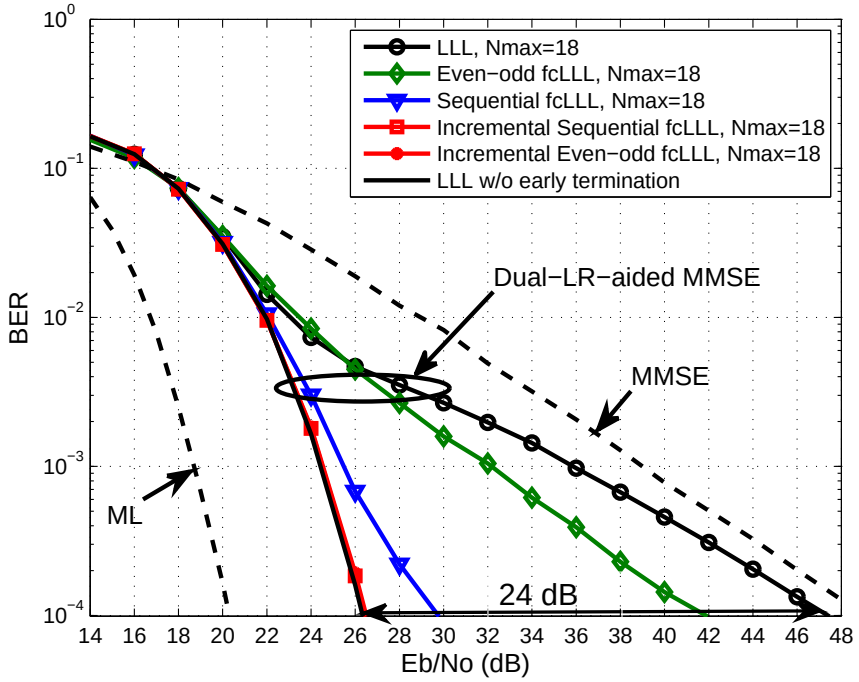
5.3.2 BER Comparisons of Different LR-aided MIMO Detectors

Fig. 5.5(a) depicts the uncoded BER performance versus the number of LLL iterations of different LR algorithms (i.e., the Incremental Sequential and Incremental Even-odd fcLLL, the Sequential fcLLL, the Even-odd fcLLL, and the LLL/ELLL) in dual-LR-aided MMSE detectors in an 8×8 MIMO system using 64-QAM. Here the sorted QR [84] is adopted since it could decrease the number of LLL iterations. The performance bound without early termination for LLL algorithm is also provided for reference. It can be seen that the two Incremental fcLLL algorithms converge fastest, and only require the smallest number of LLL iterations (i.e., around $N_{max} = 18$) to achieve the performance bound. Fig. 5.5(b) depicts the uncoded BER performance

versus SNR of different LR algorithms with $N_{max} = 18$ LLL iterations in dual-LR-aided MMSE detectors. It can be seen that the two Incremental fcLLL algorithms achieve the performance bound with $N_{max} = 18$. And the two Incremental fcLLL algorithms save up to 24dB at BER= 10^{-4} compared with other LR algorithms with $N_{max} = 18$. Similar results of difference LR algorithms can be obtained in the LR-aided MMSE-SIC detectors as shown in Fig. 5.6.

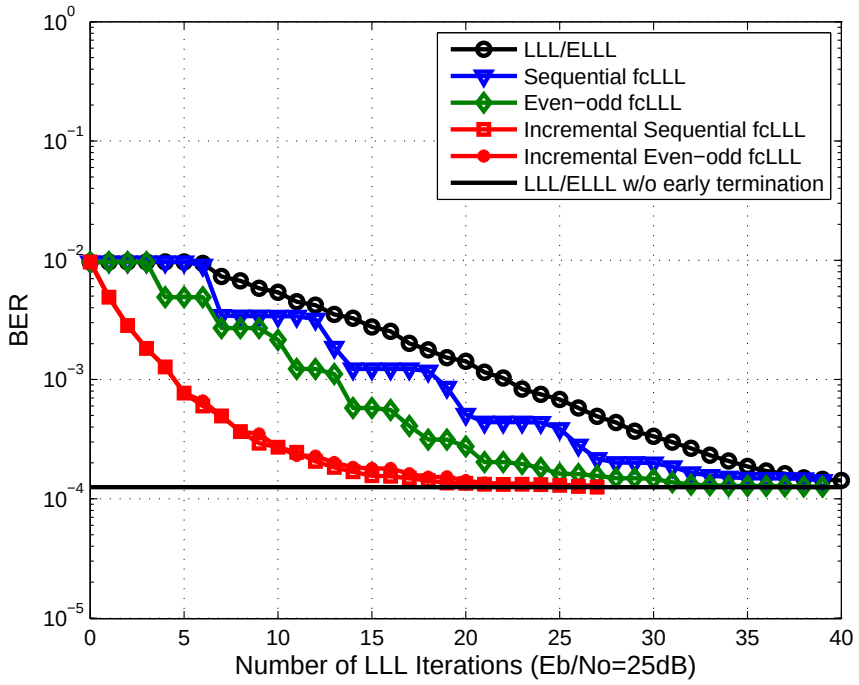


(a) BER versus LLL iterations of different LRs.

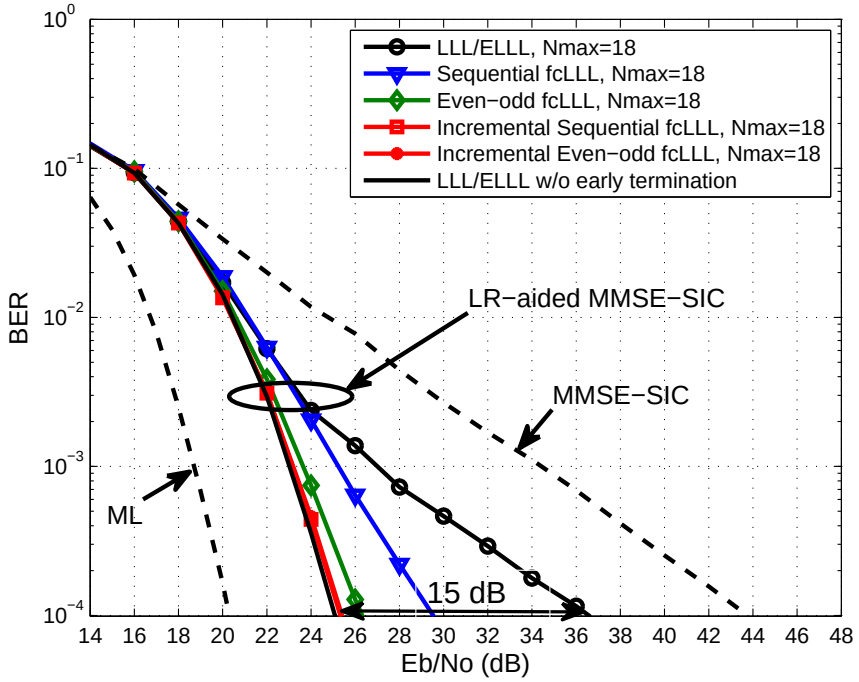


(b) BER versus SNR of different LRs with $N_{max}=18$.

Figure 5.5: BER performance of dual-LR-aided MMSE detectors with different LR algorithms in an 8×8 MIMO system using 64-QAM.



(a) BER versus LLL iterations of different LRs.



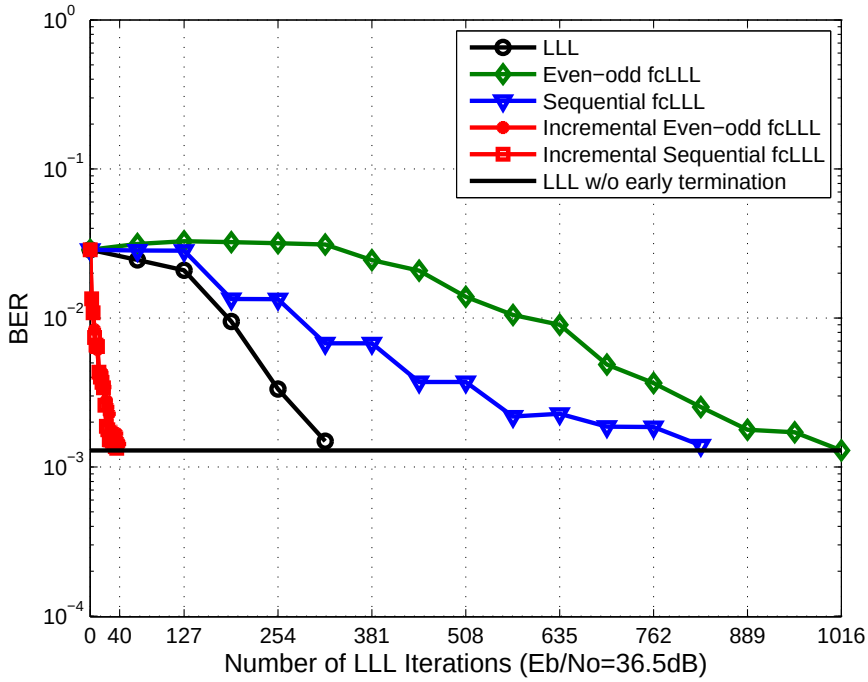
(b) BER versus SNR of different LRs with $N_{max}=18$.

Figure 5.6: BER performance of LR-aided MMSE-SIC detectors with different LR algorithms in an 8×8 MIMO system using 64-QAM.

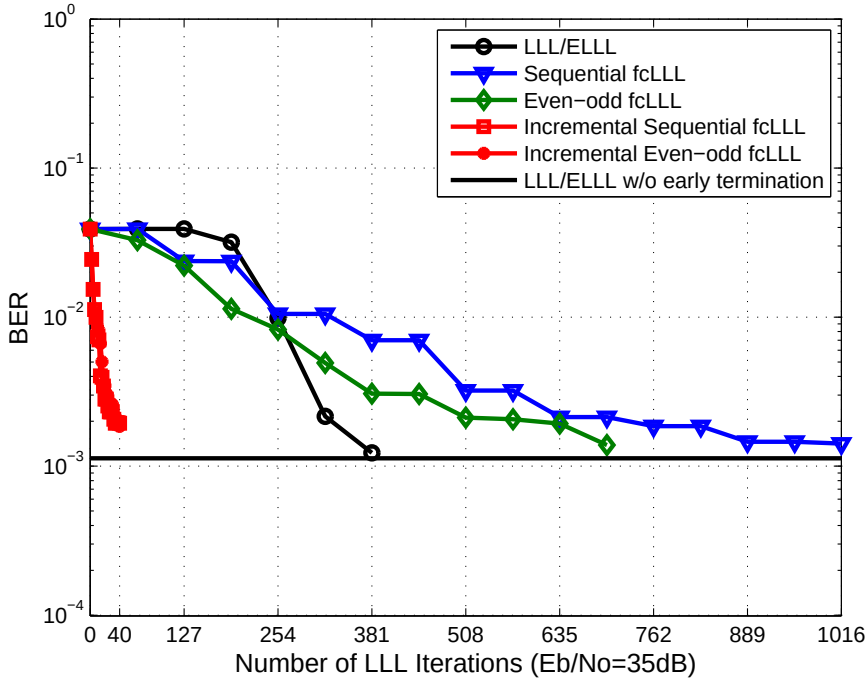
5.3.3 BER Comparisons of Different LR-aided Detectors in Large MIMO Systems

With the evolution of wireless communications, large MIMO (also known as massive MIMO) with tens or hundreds of antennas are proposed as a key technology for future fifth generation (5G) cellular network [7, 8, 28, 37, 50, 100]. It is more challenging to design high performance detectors with low complexity in large MIMO systems. In this case, lattice reduction has also attracted lots of interests [42, 59, 102] due to its desirable scaling property with polynomial complexity.

In this section, without loss of generality, we consider a large MIMO system (128×128 MIMO) with 64-QAM. Fig. 5.7 depicts the uncoded BER performance versus the number of LLL iterations of different LR algorithms in dual-LR-aided ZF and LR-aided SIC detectors. It is interesting that the Even-odd fcLLL and Sequential fcLLL algorithms need much more LLL iterations than the LLL algorithm, which indicates that Even-odd and Sequential fcLLLs may not be desirable in large MIMO systems. However, our proposed fcLLLs exhibit significant complexity advantage over others and require about 40 LLL iterations to obtain near the bound performance, which is only about 10% and 5% LLL iterations of the LLL and the Even-odd/Sequential fcLLL algorithms, respectively.



(a) Different LRs used for dual-LR-aided ZF detectors.



(b) Different LRs used for LR-aided SIC detectors.

Figure 5.7: BER performance versus number of LLL iterations of different LR algorithms in a 128×128 MIMO system using 64-QAM.

CHAPTER VI

HARDWARE-ORIENTED INCREMENTAL FCLLL ALGORITHM AND FIXED-POINT DESIGN

In this chapter, we first propose a hardware-oriented modified Incremental fcLLL algorithm for efficient hardware implementation. The proposed modified Incremental fcLLL algorithm adopts simplified size reduction and Siegel condition as well as a novel two-angle complex Givens rotation for column swap, which eliminates all computationally intensive operations (e.g., multiplication, division, norm, etc.). Then, we propose a fixed-point conversion scheme to realize the fixed-point design for the modified Incremental fcLLL algorithm. Finally, simulations demonstrate that the proposed hardware-oriented modified Incremental fcLLL algorithm with fixed-point design can maintain similar error performance as the floating-point counterpart in LR-aided MIMO detection. The contents of this chapter are based on our publications [76, 79].

6.1 Hardware-Oriented Incremental fcLLL Algorithm

The hardware-oriented Incremental fcLLL algorithm based on QR preprocessing is summarized in Table 6.1, where $IncrSeq$ indicates the iteration sequence in the Incremental fcLLL algorihm and N_{iter} indicates the fixed number of iterations. During each iteration, the algorithm performs size reduction (Lines 4-8), Siegel condition (Line 9), and possible column swap (Lines 10-13). Due to the limited value of μ_{clip} used in practice (usually 2 or 3) and the specific constant $(1 + 2^{-1} - 2^{-4})$, both size reduction and Siegel condition can be implemented by simple operations (e.g., addition, comparison, shifting, etc.). For column swap, the proposed 2-angle complex

Table 6.1: Hardware-oriented Incremental fcLLL Algorithm

Input: $\boldsymbol{Q},\ \boldsymbol{R},\ \boldsymbol{P}$ (after QR/sorted QR of $\boldsymbol{H}$: $\boldsymbol{HP}=\boldsymbol{QR}$)
Output: $\tilde{\boldsymbol{Q}}^{\mathcal{H}},\ \tilde{\boldsymbol{R}},\ \boldsymbol{T}$ ($\tilde{\boldsymbol{H}}=\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{R}}=\boldsymbol{HT}=\boldsymbol{QRT}$)

1: Initialize: $\tilde{\boldsymbol{Q}}^{\mathcal{H}}=\boldsymbol{Q}^{\mathcal{H}},\ \tilde{\boldsymbol{R}}=\boldsymbol{R},\ \boldsymbol{T}=\boldsymbol{P},\ \mu_{clip},\ N_{iter},\ IncrSeq$
2: **for** $n_{iter}=1:N_{iter}$
3: $k=IncrSeq(n_{iter})$
4: **for** $n=k-1:-1:1$ } ***simplified size reduction***
5: $\mu=\text{Quantizer}(\tilde{R}_{n,k}/\tilde{R}_{n,n})\in\{0,\pm1,\ldots,\pm\mu_{clip}\}$
6: $\tilde{\boldsymbol{R}}_{1:n,k}=\tilde{\boldsymbol{R}}_{1:n,k}-\mu\tilde{\boldsymbol{R}}_{1:n,n}$
7: $\boldsymbol{T}_{:,k}=\boldsymbol{T}_{:,k}-\mu\boldsymbol{T}_{:,n}$
8: **end**
9: **if** $|\tilde{R}_{k-1,k-1}|>(1+2^{-1}-2^{-4})|\tilde{R}_{k,k}|$ } ***simplified Siegel condition***
10: $\boldsymbol{\Theta}=\frac{1}{||\tilde{\boldsymbol{R}}_{k-1:k,k}||}\begin{bmatrix}\tilde{R}^{*}_{k-1,k} & \tilde{R}_{k,k}\\ -\tilde{R}_{k,k} & \tilde{R}_{k-1,k}\end{bmatrix}$
$=\begin{bmatrix}1 & 0\\ 0 & e^{j\theta_x}\end{bmatrix}\begin{bmatrix}\cos\theta_y & \sin\theta_y\\ -\sin\theta_y & \cos\theta_y\end{bmatrix}\begin{bmatrix}e^{-j\theta_x} & 0\\ 0 & 1\end{bmatrix}$
with $\theta_x=\tan^{-1}\left(\frac{\text{Im}(\tilde{R}_{k-1,k})}{\text{Re}(\tilde{R}_{k-1,k})}\right),\ \theta_y=\tan^{-1}\left(\frac{\tilde{R}_{k,k}}{|\tilde{R}_{k-1,k}|}\right)$ } ***two-angle CGR column swap***
11: $\tilde{\boldsymbol{R}}_{k-1:k,k-1:N_t}=\boldsymbol{\Theta}\tilde{\boldsymbol{R}}_{k-1:k,k-1:N_t}$
12: $\tilde{\boldsymbol{Q}}^{\mathcal{H}}_{k-1:k,:}=\boldsymbol{\Theta}\tilde{\boldsymbol{Q}}^{\mathcal{H}}_{k-1:k,:}$
13: exchange columns $k-1$ and k in $\tilde{\boldsymbol{R}}$ and $\boldsymbol{T}$
14: **end**
15: **end**

Givens rotation (2-angle CGR) based method adopts COordinate Rotation DIgital Computer (CORDIC) which can also be implemented by simple operations like addition, shifting, and multiplexer. The detailed design considerations of each part are elaborated as follows.

6.1.1 Simplified Size Reduction and Siegel Condition

The size reduction consists of a for loop (see Lines 4-8 of Table 6.1) with the division and rounding operation ($\mu=\lceil\tilde{R}_{k-1,k}/\tilde{R}_{k-1,k-1}\rfloor$) as well as the multiplications ($\mu\tilde{\boldsymbol{R}}_{1:n,n}$ and $\mu\boldsymbol{T}_{:,n}$), which is computationally intensive. Simulations show that most values of μ lie within a very small range [55, 56]. Therefore, we clip and quantize the values of μ within the range $[-\mu_{clip},\ \mu_{clip}]$ as shown in Line 5. This allows us to

replace the division and rounding with simple addition and comparison in hardware. Furthermore, the multiplications with μ can also be replaced with simple addition and shifting operations.

For low-complexity, we replace the Lovász condition in the LLL and fcLLL algorithms with the simplified Siegel condition in [18] as

$$|\tilde{R}_{k-1,k-1}| \leq (1 + 2^{-1} - 2^{-4})|\tilde{R}_{k,k}|, \qquad \forall\ 2 \leq k \leq N_t. \tag{6.1}$$

The implementation of (6.1) requires only simple operations (addition, shifting, comparison, and absolute value of real number).

6.1.2 Proposed Two-Angle CGR for Column Swap

The main calculations of column swap locate in Lines 10-12 of Table 6.1, which is the most computationally intensive part in the LLL/fcLLL algorithm due to the norm, division, and matrix multiplication operations. We propose a novel approach in the following to solve the complexity issue of column swap.

During the LLL iteration with index k, Lines 10-12 in column swap update the rows $k-1$ and k of matrix $\tilde{\boldsymbol{R}}$ and $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ by left multiplying the unitary matrix $\boldsymbol{\Theta}$, which is a complex Givens rotation operation. The updating in the kth column of $\tilde{\boldsymbol{R}}$ is the vectoring mode in the complex Givens rotation as

$$\begin{bmatrix} \tilde{R}'_{k-1,k} \\ \tilde{R}'_{k,k} \end{bmatrix} = \boldsymbol{\Theta} \begin{bmatrix} \tilde{R}_{k-1,k} \\ \tilde{R}_{k,k} \end{bmatrix} = \begin{bmatrix} \sqrt{|\tilde{R}_{k-1,k}|^2 + \tilde{R}_{k,k}^2} \\ 0 \end{bmatrix}. \tag{6.2}$$

This makes the upper-triangular property of $\tilde{\boldsymbol{R}}$ maintained after the following exchanging operation in Line 13. The updating of other columns in $\tilde{\boldsymbol{R}}$ and $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ corresponds to the rotation mode as

$$\begin{bmatrix} m' \\ n' \end{bmatrix} = \boldsymbol{\Theta} \begin{bmatrix} m \\ n \end{bmatrix}. \tag{6.3}$$

To make low-complexity realization, we decompose the $\boldsymbol{\Theta}$ matrix as

$$\boldsymbol{\Theta} = \frac{1}{||\tilde{\boldsymbol{R}}_{k-1:k,k}||}\begin{bmatrix} \tilde{R}^*_{k-1,k} & \tilde{R}_{k,k} \\ -\tilde{R}_{k,k} & \tilde{R}_{k-1,k} \end{bmatrix} = \begin{bmatrix} \cos\theta_y e^{-j\theta_x} & \sin\theta_y \\ -\sin\theta_y & \cos\theta_y e^{j\theta_x} \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 \\ 0 & e^{j\theta_x} \end{bmatrix}\begin{bmatrix} \cos\theta_y & \sin\theta_y \\ -\sin\theta_y & \cos\theta_y \end{bmatrix}\begin{bmatrix} e^{-j\theta_x} & 0 \\ 0 & 1 \end{bmatrix}, \tag{6.4}$$

where θ_x and θ_y are calculated as

$$\theta_x = \tan^{-1}\left(\frac{\mathrm{Im}(\tilde{R}_{k-1,k})}{\mathrm{Re}(\tilde{R}_{k-1,k})}\right),\ \theta_y = \tan^{-1}\left(\frac{\tilde{R}_{k,k}}{|\tilde{R}_{k-1,k}|}\right). \tag{6.5}$$

Based on the formulation of (6.4), the vectoring and rotation modes of the complex Givens rotation can be implemented by simple real CORDIC operations shown in Fig. 6.1, which is denoted as two-angle complex Givens rotation (two-angle CGR) since two angles (θ_x and θ_y) are calculated during the vectoring mode and utilized during the rotation mode. Since only one vector in $\tilde{\boldsymbol{R}}$ performs vectoring mode followed by rotation mode for all other vectors in $\tilde{\boldsymbol{R}}$ and $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$, there are four CORDIC cores in the proposed two-angle CGR, which means the two CORDIC cores in the vectoring mode are also used in the upper two CORDIC operations in the rotation mode in Fig. 6.1.

The proposed two-angle CGR simplifies the calculations in column swap and allows easier high-speed pipelining implementation. It operates directly for complex matrix instead of the real matrix counterpart with doubled size in [30, 55]. Thanks to the designed two-angle CGR scheme, we can achieve lower latency and higher throughput than [30, 55] in the following architecture design (see Chapters 7 and 8).

6.2 Fixed-point Design with Wordlength Optimization

For the fixed-point design of the proposed modified Incremental fcLLL algorithm, all data and arithmetic operations are converted from floating-point version to fixed-point version based on the actual hardware architecture. We combine the heuristic

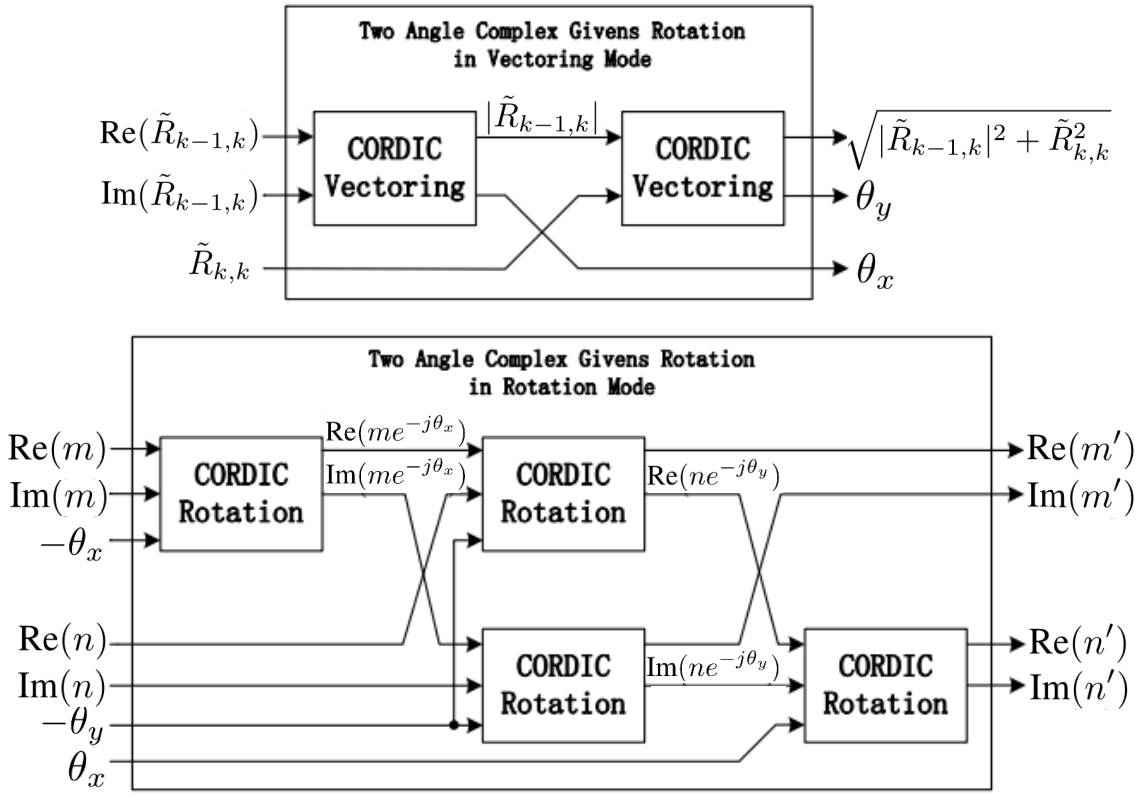


Figure 6.1: Proposed two-angle complex Givens rotation for column swap.

methods in [79] and [61] to complete our fixed-point design. That is, the parameters/modules that decide the major hardware cost are optimized first [61] followed by wordlength optimization for each variable [79]. The detailed procedure for fixed-point design is summarized as follows:

1. Decide the iteration number of the modified Incremental fcLLL algorithm. This parameter is the most important one, since the hardware cost is approximately linear to the iteration number in our designed two architectures.

2. Decide the iteration number of the CORDIC operations in the two-angle CGR. Later in Table 8.1 we will show that the module of the two-angle CGR for column swap has the highest hardware cost among all modules. Meanwhile, the hardware cost of the two-angle CGR is approximately linear to the iteration

number of the CORDIC operations.

3. Decide the threshold μ_{clip} in size reduction. The hardware cost of size reduction is mainly affected by the value of μ_{clip} based on the design in Section 6.1.1, and the size reduction module has the second highest hardware cost among all modules as shown in later Table 8.1.

4. Decide the wordlength for each variable by considering the two-angle CGR module first followed by size reduction module and then followed by other modules.

For the last step of the fixed-point design, the wordlength optimization of each variable can be performed by analysis or simulation [14]. The former is usually conservative and difficult to analyze in nonlinear and unsmooth operations [46]. So here we adopt the latter. However, the simulation-based scheme is an NP-hard combinatorial problem [15], which leads to exceedingly long simulation time. To avoid this problem, we propose a scheme by combining the *Heuristic procedure* and *Max-1 bit procedure* summarized in [11]. It utilizes the small number of iterations from the *Heuristic procedure* and the minimum total bit-width from *Max-1 bit procedure*. The proposed wordlength optimization procedure is depicted in Fig. 6.2 and summarized as follows:

- **Range Estimation**: Determine the minimum integer wordlengh (iwl) to prevent overflow and underflow, which can be obtained by examining the histograms of each fixed-point variables under large data simulation.
- **Precision Estimation**: Find the minimum fraction wordlengh (fwl) so that the performance degradation under quantization noise can be retained within the tolerated error metric. This step consists of the following three sub-steps:
  § Find fwl_{min}: Obtain the smallest fwl of each variable that satisfies the tolerated error metric when the fwl of all other variables are large enough (here we use 32 bits).

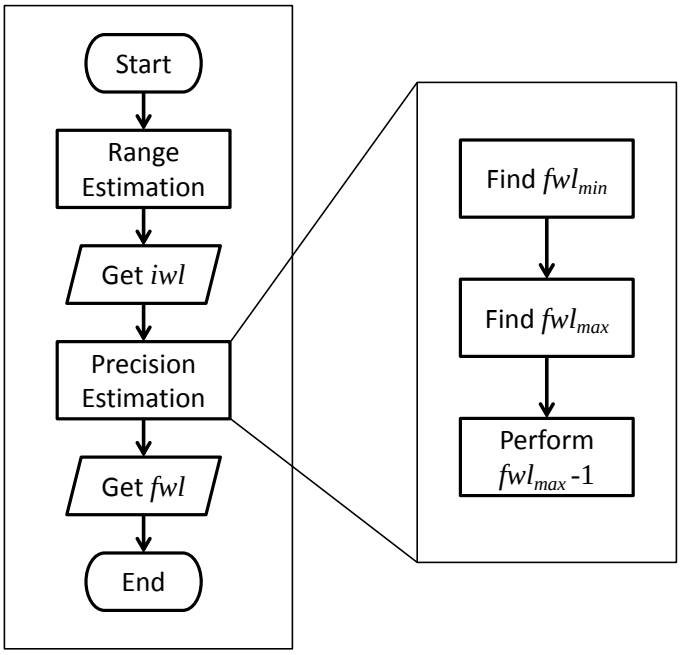


Figure 6.2: The proposed wordlength optimization procedure.

§ Find fwl_{max}: Increase fwl_{min} of all variables simultaneously with 1 bit as step size until the tolerated error metric is met. The updated values of fwl_{min} are the corresponding fwl_{max}.

§ Perform $fwl_{max}-1$: Record the BER by reducing the fwl_{max} of each variable 1 bit while keeping the fwl_{max} of all other variables unchanged. The fwl_{max} of the variable with the best BER performance is updated as $fwl_{max}-1$. This process is repeated until the tolerated error metric is not satisfied to get the final optimized fwl.

To obtain the aforementioned parameters and the wordlength of each variable in the modified Incremental fcLLL algorithm, we develop a complete fixed-point C++ model which is bit-accurate with the corresponding Verilog HDL code. The final

results adopted in hardware for the iteration number of the modified fcLLL, the iteration number of the CORDIC, and the threshold of μ_{clip} are 5, 8, and 2, respectively. The wordlength of the fixed-point configurations (format: [integer bits, fraction bits]) for $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$, $\tilde{\boldsymbol{R}}$, and $\boldsymbol{T}$ elements are [2,14], [5,11], and [8,0], respectively.

To evaluate the fixed-point performance of the hardware-oriented Incremental fcLLL algorithm, we compare the uncoded BER results between fixed-point and floating point Incremental fcLLL algorithm in the complex LR-aided MMSE K-best detector for a 4×4 MIMO system with 64-QAM as shown in Fig. 6.3, where sorted QR is adopted for all LRs to reduce iterations [84] and $K = 3$ candidates are used in the K-best detector. It can be seen that original and modified Incremental fcLLLs achieve almost the same performance as the ML detector, and the performance loss of the fixed-point design is only around 0.27 dB at BER=10^{-4} compared to the floating-point counterpart.

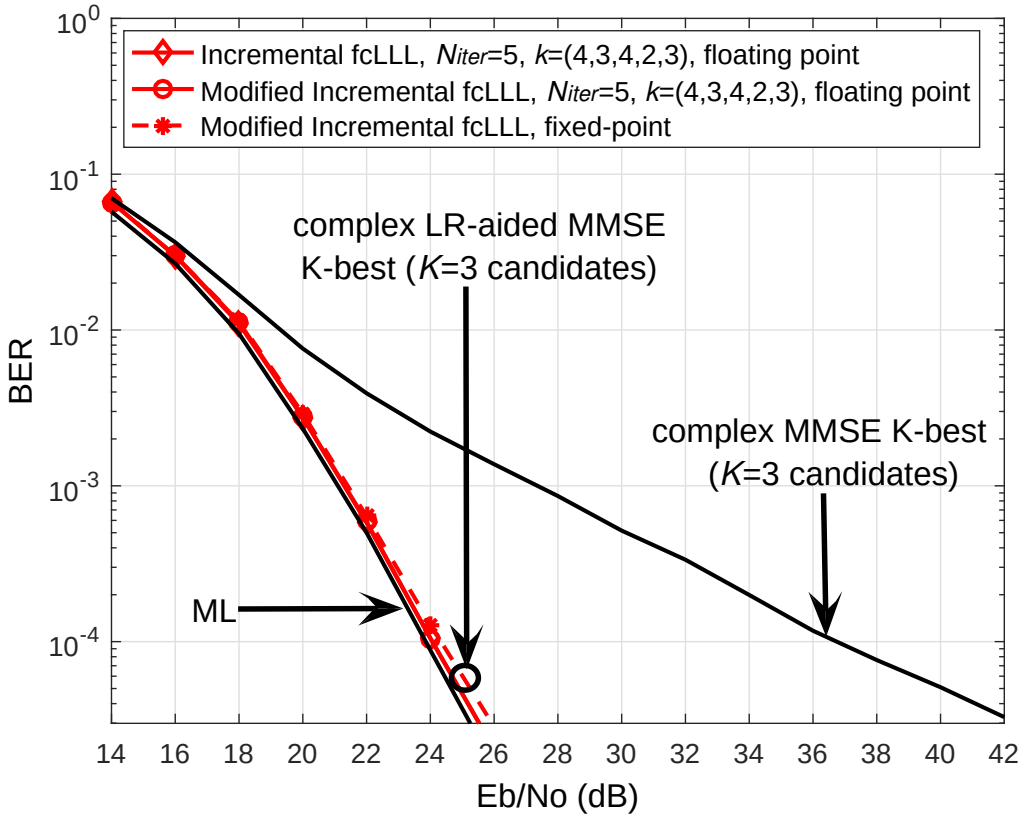


Figure 6.3: BER comparison between fixed-point and floating point of the hardware-oriented modified Incremental fcLLL algorithm in the complex LR-aided MMSE K-best detector ($K = 3$ candidates) for a 4×4 MIMO system with 64-QAM.

CHAPTER VII

ITERATIVE HARDWARE IMPLEMENTATION OF INCREMENTAL FCLLL ALGORITHM

In this chapter, we present a low-complexity iterative architecture to implement the hardware-oriented Incremental fcLLL algorithm proposed in Chapter 6. Compared to the pipelined counterpart which will be discussed in next chapter, the proposed iterative architecture provides a tradeoff between throughput and complexity with much lower utilization of hardware resources. First, we design an efficient architecture, such that all the modules are iteratively time-multiplexed among LLL iterations. Second, each module in the datapath is designed with low complexity and high clock frequency. Third, the properties of dual-port memory are exploited to simplify the operation of data exchange and transfer without affecting the minimum-achievable processing latency. Finally, the implementations on Xilinx Virtex-4/5/7 devices demonstrate much lower utilization of FPGA resources while still achieving comparable throughput compared to the existing FPGA solutions. The contents of this chapter are based on our publication [78].

7.1 Proposed Iterative Hardware Architecture

In this section, we develop an iterative architecture of the Incremental fcLLL for 4×4 MIMO systems based on Table 6.1. The high-level architecture is depicted in Fig. 7.1, which is time-multiplexed among iterations. The main datapath contains the modules of Size Reduction, Siegel Condition, 2-angle CGR based Column Swap, and three dual-port SRAM memories to store intermediate values of $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$, $\tilde{\boldsymbol{R}}$ and $\boldsymbol{T}$ matrices. The control path locates in the Global Controller module, which is

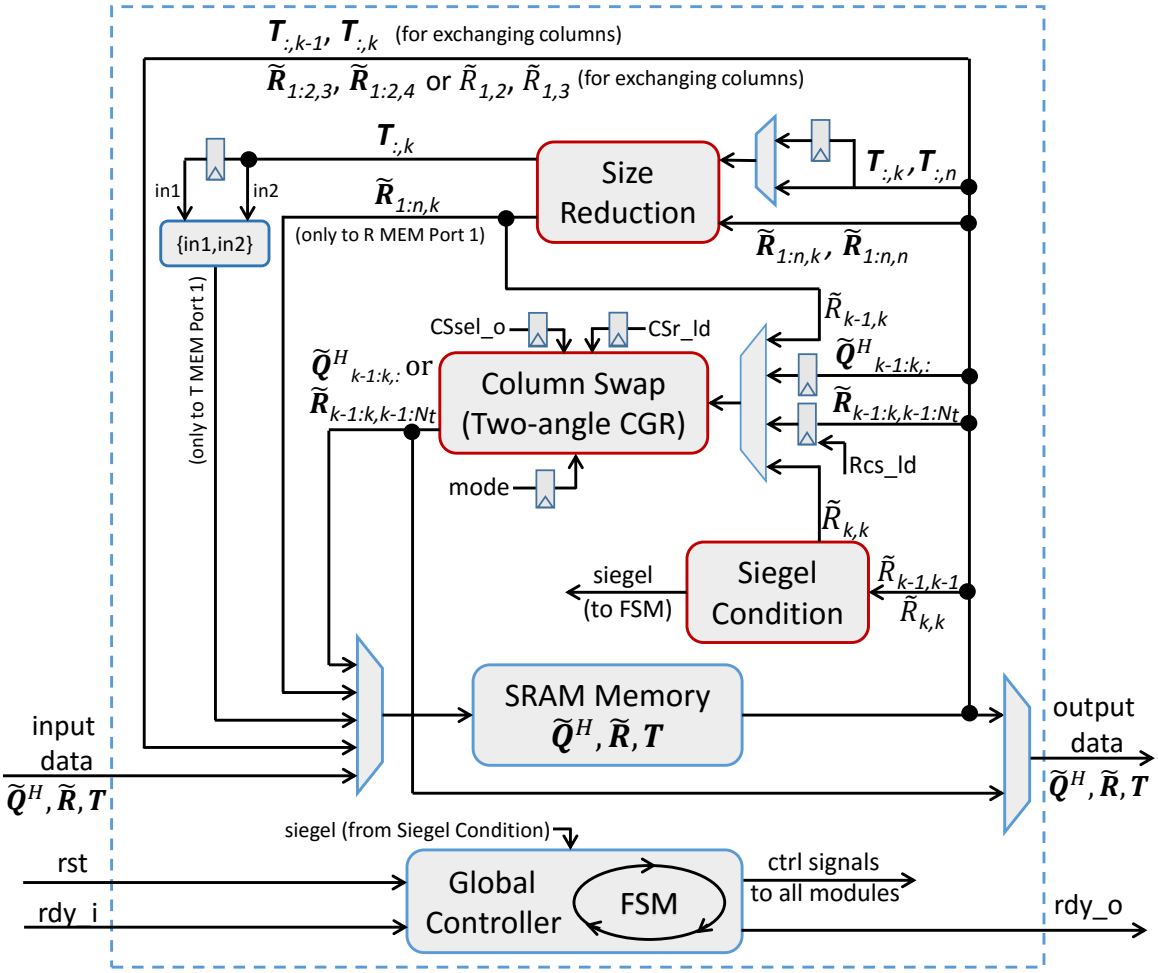


Figure 7.1: Proposed iterative architecture of the Incremental fcLLL algorithm.

implemented by a finite state machine (FSM). The functionalities of the datapath and control path are elaborated in the following parts.

7.1.1 Datapath Design

The designed architecture of the simplified size reduction is summarized in Fig. 7.2, which is based on $\mu_{clip}=2$ according to the fixed-point design in previous chapter. Following the formulation of the simplified size reductio in Section 6.1.1, we adopt the similar implementation as in [55], which can be mapped to simple operations like addition, shifting and comparison. One difference here is that we apply retiming to the μ calculation block to change the needed cycles from 1 to 2 for higher clock

frequency (see the lower part of Fig. 7.2). Otherwise, this part would be the critical path with the longest combinational delay in the whole architecture which leads to lower throughput. Note that the outputs of the μ calculation block are registered with load input u_ld, such that these outputs are updated only when the u_ld signal is asserted. Based on Lines 4-8 of Table 6.1, it can be seen that the u_ld signal needs to be asserted $k-1$ times in one fcLLL iteration. Another difference is that we add an extra size reduction block for the updating of $\boldsymbol{T}$, which means that two blocks of the size reduction element (SRE) are used for either $\tilde{\boldsymbol{R}}$ or $\boldsymbol{T}$ data such that each SRE only processes the real or imaginary part of a complex matrix entry. By doing so, the updating of one matrix entry in $\tilde{\boldsymbol{R}}$ or $\boldsymbol{T}$ requires only 1 cycle. The third difference is that we insert two D flip-flops (DFFs) before each SER for data synchronization. This is because the input data is continually streaming through the datapath and the size reduction can be performed as soon as the calculated μ is available.

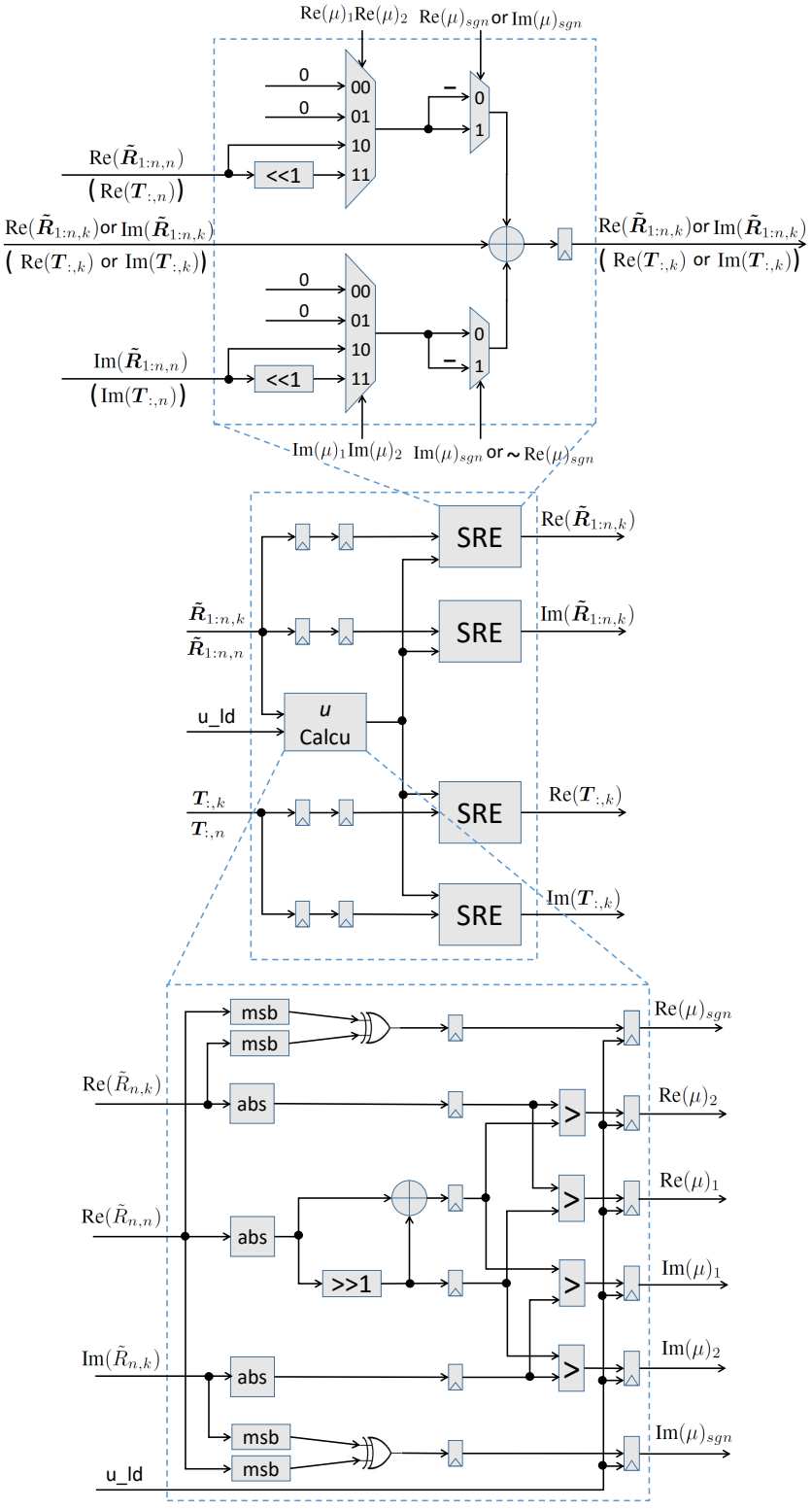


Figure 7.2: Architecture of the size reduction module.

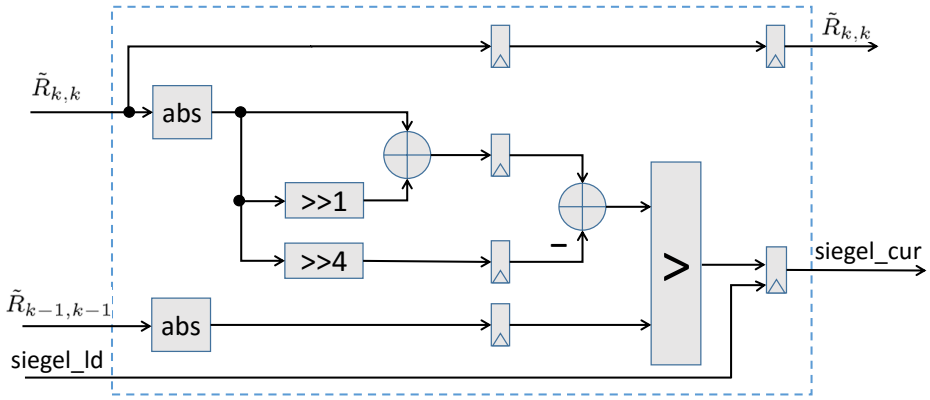


Figure 7.3: Architecture of the Siegel condition module.

Compared to the size reduction module, the architecture design of the simplified Siegel condition is relative straightforward. Based on the Eq. (6.1), the Siegel condition evaluation can be implemented only by two shifting followed by two addition and one comparing operations as shown in Fig. 7.3, where we also apply retiming for higher clock frequency, resulting 2 cycles processing time. Since the Siegel condition only needs to be calculated once in a fcLLL iteration and its result will be used in the following column swap and data transfer operations, the Siegel output of current iteration *siegel_cur* is registered with load input *siegel_ld*. Note that we also output the original input $\tilde{R}_{k,k}$ synchronized with the *siegel_cur* signal, such that the $\tilde{R}_{k,k}$ element of the vectoring operation in column swap is ready (see Eq. (6.2)) once the Siegel condition *siegel_cur* is known. Another goal of this design is to avoid SRAM memory read for $\tilde{R}_{k,k}$ at the vectoring operation for global timing optimization.

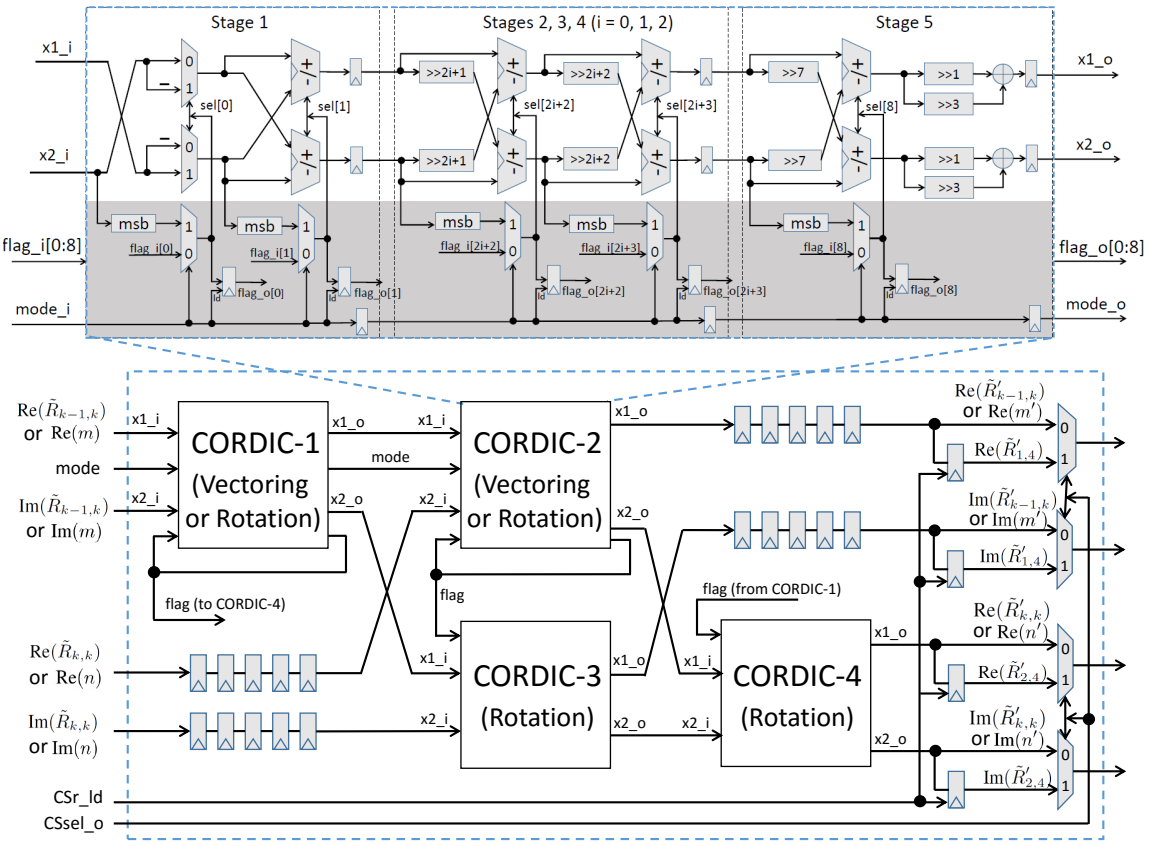


Figure 7.4: Architecture of the 2-angle CGR architecture for column swap.

For the 2-angle CGR based Column Swap module, the architecture is depicted in Fig. 7.4. Compared to that of the pipelined architecture which will be discussed in next chapter, the bypass circuits (output = input) are completely removed to save hardware resources and reduce the complexity of each CORDIC core. This is because we can disable the $\tilde{\boldsymbol{R}}$ and $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ memories during the output period of the column swap if the Siegel condition is false, while in this case the pipelined architecture must transfer the original data (i.e., output = input for column swap) to the next stage. Meanwhile, we do not need to store the vectoring output $\tilde{R}'_{k-1,k}$ and $\tilde{R}'_{k,k}$ in temporary registers due to the different timing schedule in our iterative architecture, which also reduces the number of output ports in Column Swap module by half. Finally, all the input signals from the controller and the input $\tilde{\boldsymbol{R}}$ and $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ data from memories are

registered for better clock frequency as shown in Fig. 7.1, which would not affect the best-achievable processing latency by careful adjusting the timing schedule.

For the detailed functionality of the 2-angle CGR in Fig. 7.4, the input data $\tilde{R}_{k-1,k}$ and $\tilde{R}_{k,k}$ are used for vectoring operation while the input data m and n indicate the other $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ or $\tilde{\boldsymbol{R}}$ elements used for rotation operation. Four basic real-valued CORDIC cores are constructed in the architecture. The CORDIC-1/2 works in either vectoring or rotation mode while the CORDIC-3/4 works only in rotation mode. Each CORDIC core is unrolled into pipelined 5 stages so the processing latency is 5 cycles, and the total processing latency of the Column Swap module is 15 cycles. The detailed pipelined design inside the CORDIC core is depicted in the upper part of Fig. 7.4. Each CORDIC core consists of 5 pipelined stages with 5 cycles latency, so the total processing latency of the two-angle CGR module is 15 cycles. The left half of Stage 1 is designed to extend the CORDIC rotation angles from $[-\pi/2, \pi/2]$ to $(-\pi, \pi]$ by an initial $\pm\pi/2$ rotation as follows [3]

$$\begin{cases} flag\_o[0] = \begin{cases} -sign(x2\_i) & \text{in vectoring mode} \\ flag\_i[0] & \text{in rotation mode} \end{cases}, \\ x_0 = -flag\_o[0] \cdot x2\_i, \\ y_0 = flag\_o[0] \cdot x1\_i. \end{cases}$$

The right half of Stage 1, the middle 3 stages, and the left half of Stage 5 correspond to 8 CORDIC iterations which is determined based on previous fixed-point simulations. The corresponding $N = 8$ CORDIC operations with $n = 0, 1, ..., N-1$ are

$$\begin{cases} flag\_o[n+1] = \begin{cases} -sign(y_n) & \text{in vectoring mode} \\ flag\_i[n+1] & \text{in rotation mode} \end{cases}, \\ x_{n+1} = x_n - y_n \cdot flag\_o[n] \cdot 2^{-n}, \\ y_{n+1} = y_n + x_n \cdot flag\_o[n] \cdot 2^{-n}. \end{cases}$$

The right half of Stage 5 is adopted to approximate the scaling factor in CORDIC as in [55]

$$\begin{cases} x1\_o = x_N \cdot (1/\prod_{n=0}^{N-1}\sqrt{1+2^{-2n}}) \approx x_N \cdot (2^{-1}+2^{-3}) \\ x2\_o = y_N \cdot (1/\prod_{n=0}^{N-1}\sqrt{1+2^{-2n}}) \approx y_N \cdot (2^{-1}+2^{-3}) \end{cases}$$

To improve efficiency, the rotation directions in CORDIC are only calculated in CORDIC-1/2 in vectoring mode and saved in $flag\_o$ by registers for latter use in rotation mode. That is why the input and output of the $flag$ signal are connected in CORDIC-1/2 as shown in Fig. 7.4. For CORDIC-3/4 which only operates in rotation mode, the shaded area in CORDIC core is removed to save complexity. In this case, the signal $flag\_i$ is directly connected to the signal sel to guide the rotation direction. Note that the retiming choice by chaining two CORDIC iterations together is decided by also considering the clock frequency of other modules such that the final throughput of the whole system is maximized.

For the data storage of $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$, $\tilde{\boldsymbol{R}}$ and $\boldsymbol{T}$ matrices in each pipelined fcLLL unit, we adopt embedded SRAM memory instead of registers as in [30, 55]. Compared to registers, embedded SRAM memory can significantly reduce storage area at the cost of limited number of parallel ports, so the required timing schedule is more stringent to achieve high throughput. Since our design is based on Xilinx FPGA, we adopt the embedded dual-port block RAM, where data can be written to (or read from) either or both ports. To exploiting its potential, the block RAM is configured as "read before write" mode [87], such that when the input data is being written into a address, the old data in this address appears on the output latches. This property is fully utilized for the data transfer between pipelined fcLLL units which will be discussed later. Considering the fixed-point design in Section 6.2 and the fact that the maximum available capacity of one memory cell is 36 bits in the Xilinx FPGA block RAM [87], we set the capacity of memory cell as 32 bits for all $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$, $\boldsymbol{R}$, and $\boldsymbol{T}$ RAMs, such that each memory cell stores one matrix entry for $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ and $\boldsymbol{R}$ matrices

while two matrix entries for $\tilde{\boldsymbol{T}}$. Therefore, three block RAMs are required in each pipelined fcLLL units, which are configured as 16×32 bits, 16×32 bits, and 8×32 bits RAMs for $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$, $\tilde{\boldsymbol{R}}$, and $\boldsymbol{T}$ matrices, respectively. Note that the design that each memory cell stores two matrix entries for $\tilde{\boldsymbol{T}}$ is also utilized to save memory operations and improve throughput. With this design for the memory cell in the dual-port block RAM, one-time memory operation can read/write at most two complex numbers for $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ or $\boldsymbol{R}$ matrices, or four complex numbers for $\boldsymbol{T}$ matrix.

7.1.2 Control Path Design, Data Exchange, and Transfer

The FSM based Global Controller module in Fig. 7.1 is responsible for the timing schedule, which is summarized in Fig. 7.5 for each iteration index k. Let us first consider the schedule of execution units. Since the processing latency of the Column Swap module is larger than others, its execution has the highest priority for throughput optimization. The first operation in column swap of each iteration is the vectoring operation of $\tilde{R}_{k-1,k}$ and $\tilde{R}_{k,k}$, where the $\tilde{R}_{k-1,k}$ is the updated value after size reduction. Thus, from cycle 1 on, the two elements $\tilde{R}_{k-1,k}$ and $\tilde{R}_{k-1,k-1}$ are fetched from $\tilde{\boldsymbol{R}}$ memory for size reduction, and the updated output $\tilde{R}_{k-1,k}$ is used for vectoring after 4 cycles period (1 cycle for RAM read, 2 cycles for initial u calculation, and 1 cycle for $\tilde{R}_{k-1,k}$'s size reduction). At cycle 2, two elements $\tilde{R}_{k-1,k-1}$ and $\tilde{R}_{k,k}$ are fetched for Siegel Condition module, where the $\tilde{R}_{k,k}$ is also used for the vectoring operation later. At cycle 5, the vectoring operation in column swap is performed, followed by the rotation operations for the other $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ and $\tilde{\boldsymbol{R}}$ elements, ending up with 26 cycles for iteration $k = 2$ or 3 and 25 cycles for iteration $k = 4$ to finish the data updating in $\tilde{\boldsymbol{R}}$ memory. Note that the last updated two elements $\tilde{R}'_{1,4}$ and $\tilde{R}'_{2,4}$ at iteration $k = 2$ in column swap are temporally stored in registers (see the rightmost part in Fig. 7.4) to achieve 26 cycles latency. These two elements are then fetched and transferred to $\tilde{\boldsymbol{R}}$ memory later at cycle 4 of the following iteration $k = 3$. Also note that the two

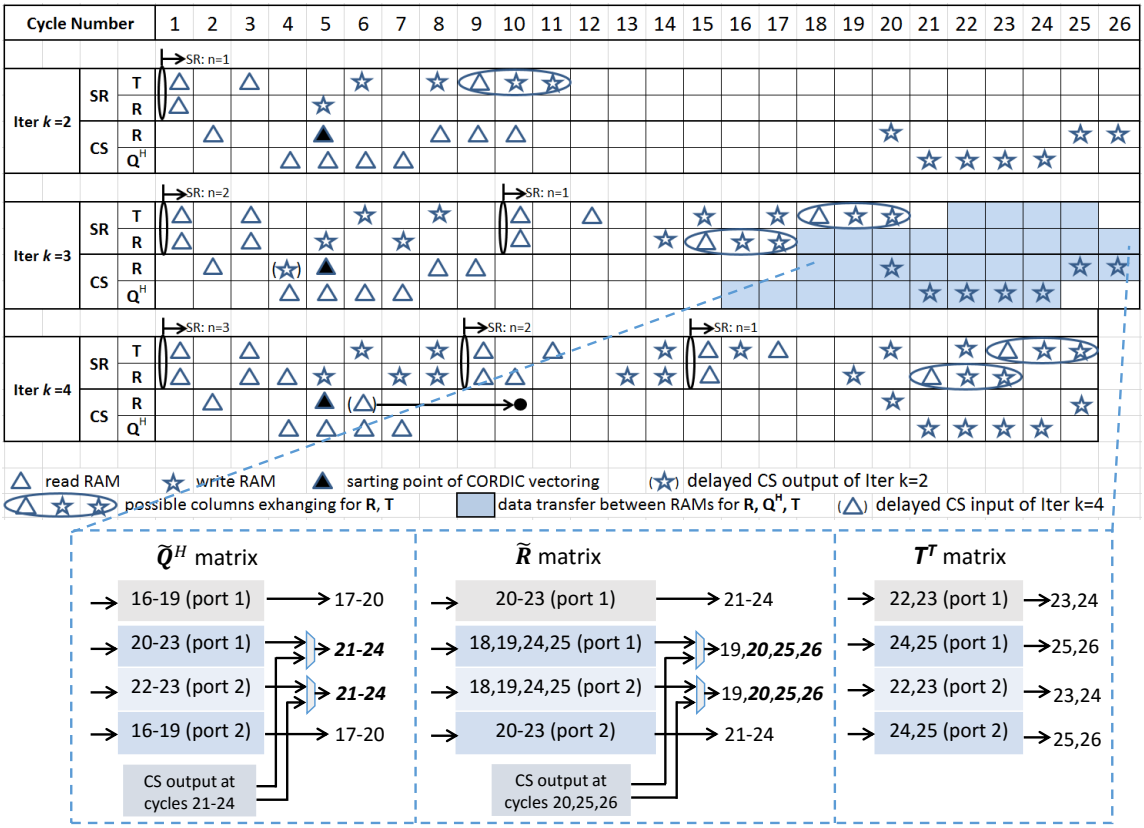


Figure 7.5: The timing schedule for data operations and transferring.

elements $\tilde{R}_{3,3}$ and $\tilde{R}_{4,3}$ at iteration $k = 4$ are fetched at cycle 6, temporally stored in registers, and used until cycle 10 for rotation operation, to achieve 25 cycles processing latency at iteration $k = 4$ without read/write conflict in $\tilde{\boldsymbol{R}}$ memory. After the column swap is scheduled to obtain minimum-achievable latency, the rest size reduction is then scheduled to utilize the available cycles which is summarized in Fig. 7.5.

Our Incremental fcLLL implementation adopts 5 iterations, i.e., $k = (4, 3, 4, 2, 3)$, based on fixed-point simulations. Thus, the minimum latency we can achieve is 128 (=25*2+26*3) cycles. Next, we consider the data exchange and transfer by exploiting the unused cycles as well as the properties of the FPGA memories to achieve this minimum-latency constraint. We adopt embedded dual-port block RAM memories in Xilinx FPGA, where data can be read/written at either or both ports. Meanwhile,

the "read-before-write" mode is configured such that old data would be transferred to the output port when new data are being written into memory. This property is utilized for both column exchange at each iteration (see Line 13 of Table 6.1) and data transfer at the final iteration $k = 3$. For the column exchange of $\tilde{\boldsymbol{R}}$ or $\boldsymbol{T}$, only 3 cycles (1 read cycle followed by 2 write cycles) are needed at each iteration as indicated by the ellipses in Fig. 7.5. Note that these 3 cycles are not needed for $\tilde{\boldsymbol{R}}$ at iteration $k = 2$, since the column swap at $\tilde{\boldsymbol{R}}_{1:2,1:2}$ is already achieved at the output of Column Swap module while the data at $\tilde{\boldsymbol{R}}_{3:4,1:2}$ are just zeros. For the data transfer at the final iteration $k = 3$, new data are coming into memories while the updated data are coming out of memories for $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$, $\tilde{\boldsymbol{R}}$, and $\boldsymbol{T}$ matrices. The corresponding occupied cycles are the shaded part in Fig. 7.5. The detailed input/output cycle indices of the memory write/read for each matrix are also provided in the lower part of Fig. 7.5, where part of the data transfer for $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ and $\tilde{\boldsymbol{R}}$ may come from the output of Column Swap based on the Siegel signal from FSM. Finally, the complete timing schedule in Fig. 7.5 indicates that the minimum latency with 128 cycles is achieved.

7.2 Implementation and Comparison

To evaluate the proposed iterative architecture, we implemented the architecture in Verilog under the FPGA RTL design flow consisting of Synopsys Synplify 2012 for synthesis, Xilinx ISE 14.7 for place-and-rout (PAR), and Xilinx ISim for verification. Next, our implementation results on Xilinx Virtex-4/5/7 FPGAs are compared to the recent published LLL hardware results supporting 4×4 MIMO with similar Virtex-4/5 FPGAs, i.e., the LLL with Siegel condition (S-LLL) [18], the reverse Siegel LLL (RS-LLL) [9], and the constant-throughput LLL (CT-LLL) [30]. The final results are summarized in Fig. 7.6 and Table 7.1 based on the performance in MIMO detection and FPGA implementations, respectively.

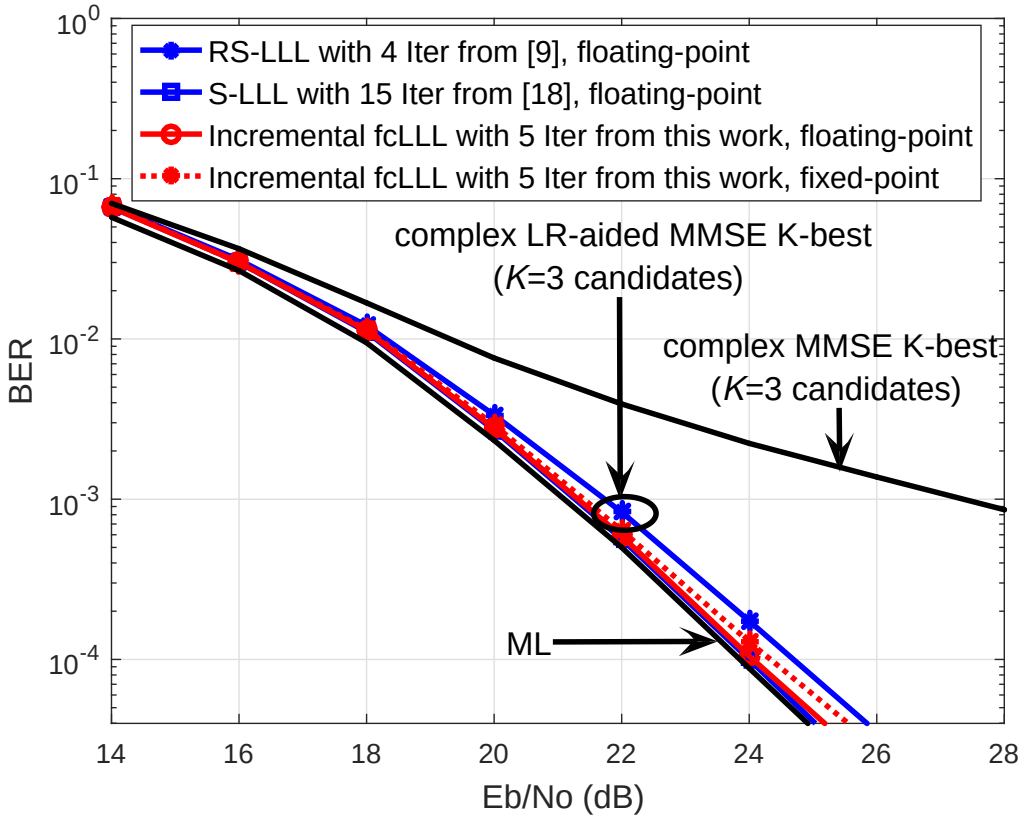


Figure 7.6: BER comparison of LLL variants in FPGA realizations in the complex LR-aided MMSE K-best detector for a 4×4 MIMO system with 64-QAM.

7.2.1 Performance Comparison in MIMO Detection

We consider the complex LR-aided MMSE K-best detection [79] in a 4×4 MIMO system with 64-QAM, where sorted QR [84] is used for LLL variants to reduce iterations and $K = 3$ candidates are adopted in K-best detection. In Fig. 7.6, the uncoded BER performances of RS-LLL, S-LLL, and Incremental fcLLL are depicted, where the number of iteration comes from the corresponding FPGA solution. First, we compare the floating-point results (solid lines) which represents the best-achievable BER performances of different FPGA solutions. The LR-aided detection with either S-LLL or Incremental fcLLL can achieve similar error performance as the ML detection, with less than 0.2 dB performance loss at BER=10^{-4}. For RS-LLL, there is around 0.8 dB performance loss at BER=10^{-4}, mainly due to the small number of iteration used in

RS-LLL. We omitted the BER curve of the CT-LLL, since it is originally designed for real instead of complex MIMO detection. But its BER performance is at most near the ML detection (i.e., similar to the S-LLL or Incremental fcLLL). Last, the fixed-point performance of our proposed design is also provided (dashed line), and the performance loss is only about 0.27 dB at BER=10^{-4} compared to the floating-point version.

7.2.2 Performance Comparison of FPGA Implementations

Table 7.1 summarizes the recent FPGA results of LLL variants for 4×4 MIMO systems. Since all LLL variants have similar Virtex-4/5 FPGA and *throughput = clock frequency ÷ (cycles/matrix)*, we focus on the throughput and FPGA resources for discussion. First, compared to the S-LLL with Virtex-4/5 [18], the throughput of our design is between the average and worst cases of [18], but our design has the advantages of fixed throughput and much lower FPGA utilization (less than half slices of [18] under the same Virtex-5). Second, compared to the RS-LLL with the same Virtex-4 [9], our design has the advantage of fixed throughput, although our throughput is lower than the average throughput of [9]. But the design in [9] consumes 2.7-fold slices compared to ours, and uses 16 FPGA multipliers while our design does not use FPGA multipliers. Furthermore, large BER performance loss exists in [9] as shown in Fig. 7.6. Third, compared to the CT-LLL implementation with Virtex-4 in [30], our design exhibits both throughput and complexity advantages, i.e., our design has over 55% throughput improvement with only around 16% slices of [30]. Finally, our implementation on Virtex-7 (XC7VX485T-3) is also provided, where the throughput is further increased to 2.2 million matrices per second with only around 1% slices utilization.

Table 7.1: Comparison of FPGA Implementations of the LLL Variants for 4×4 MIMO systems.

Reference		TCAS-I'2011 [18]		ISCAS'2010 [9]	TCAS-II'2011 [30]	This work		
LR Algorithm		S-LLL		RS-LLL	CT-LLL	Iterative Incremental fcLLL		
FPGA Platform Device Number		Virtex-4 XC4VLX80-12	Virtex-5 XC5VLX110-3	Virtex-4 XC4VLX160-12	Virtex-4 XC4VLX60-n.a.	Virtex-4 XC4VLX160-12	Virtex-5 XC5VLX110-3	Virtex-7 XC7VX485T-3
Clock Frequency (MHz)		173	206	79	8.7	187	225	273
Cycles/Matrix		49^{avg}, 447^{worst}		14^{avg}, n.a.$^{\text{worst}}$	12^{fixed}	128^{fixed}		
Throughput (MMat/s)		3.53^{avg}, 0.39^{worst}	4.20^{avg}, 0.46^{worst}	5.6^{avg}, n.a.$^{\text{worst}}$	0.73^{fixed}	1.46^{fixed}	1.76^{fixed}	2.13^{fixed}
FPGA Resources	Slices (ratio)	3,571 (9.96%)	1,758 (10.17%)	4,805 (7.11%)	11,330 (42.56%)	1,765 (2.61%)	870 (5.09%)	842 (1.11%)
	Block RAMs	n.a.	n.a.	0	n.a.	3	3	3
	Multipliers	4	4	18	n.a.	0	0	0

Note: n.a. indicates "not available"; avg indicates "average".

7.3 Conclusion

In this chapter, we proposed an iterative architecture of the Incremental fcLLL algorithm for LR-aided MIMO detection. Due to the efficient iterative architecture, novel 2-angle CGR for column swap, and the carefully designed timing schedule, our implementation has much lower hardware utilization than the existing FPGA solutions while still achieving comparable throughput performance.

CHAPTER VIII

PIPELINING HARDWARE IMPLEMENTATION OF INCREMENTAL FCLLL ALGORITHM

In this chapter, we present a high-throughput pipelined VLSI implementation of the hardware-oriented Incremental fcLLL algorithm proposed in Chapter 6. Although the basic size reduction and Siegel condition modules from the iterative architecture in the previous chapter are reused, the two-angle CGR module for column swap is updated to support pipelined implementation. Furthermore, the modules cooperation for different matrices are more complex in the pipelined architecture, which will be elaborated in this chapter. Finally, the timing schedule specifically optimized for the pipelined structure is designed to achieve maximum throughput. The implementations on Xilinx Virtex-4/5/7 FPGA devices demonstrate a processing period of 26 cycles per matrix, resulting in throughput up to 9.9 million matrices per second, which has much higher throughput than state-of-the-art FPGA implementations, and similar even better throughput than the recent ASIC/ASIP implementations. The contents of this chapter are based on our publications [76, 77].

8.1 Proposed Pipelined Hardware Architecture

The high-level pipelined hardware architecture of the proposed hardware-oriented Incremental fcLLL algorithm for 4×4 MIMO systems is shown in Fig. 8.1. The architecture consists of a global controller and five pipelined fcLLL units corresponding to five iterations, i.e., $k = (4, 3, 4, 2, 3)$. Each fcLLL unit contains three datapaths (size reduction, Siegel condition, and the two-angle CGR for column swap), three dual-port SRAM memories (for $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$, $\tilde{\boldsymbol{R}}$, and $\boldsymbol{T}$), and a local controller (for timing

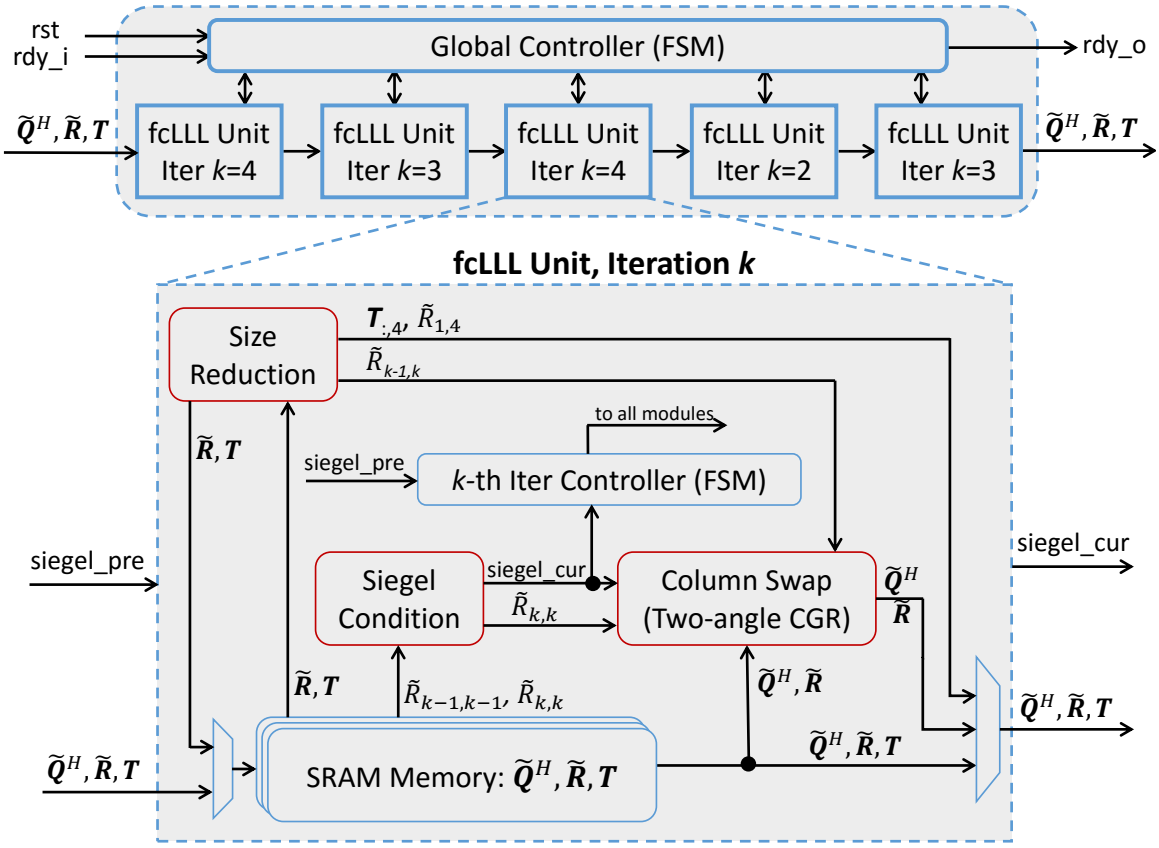


Figure 8.1: Proposed high-level pipelined architecture of the modified Incremental fcLLL algorithm for 4×4 MIMO systems.

schedule). Besides the basic size reduction and Siegel condition modules from the iterative architecture in the previous chapter, all other modules and operations are different. Next, we explain the details for the pipelined hardware architecture.

8.1.1 Architecture Design of the Two-Angle CGR Module for Column Swap

The updating of $\tilde{\boldsymbol{R}}$ and $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ matrices in column swap is implemented using the proposed two-angle CGR scheme. The architecture design adopted in this chapter is depicted in Fig. 8.2, which contains four CORDIC cores with pipelined design inside. The CORDIC-1 and CORDIC-2 work in either vectoring mode or rotation mode while CORDIC-3 and CORDIC-4 only work in rotation mode as discussed in the previous Section 6.1.2. Besides, each CORDIC core also contains bypass circuits

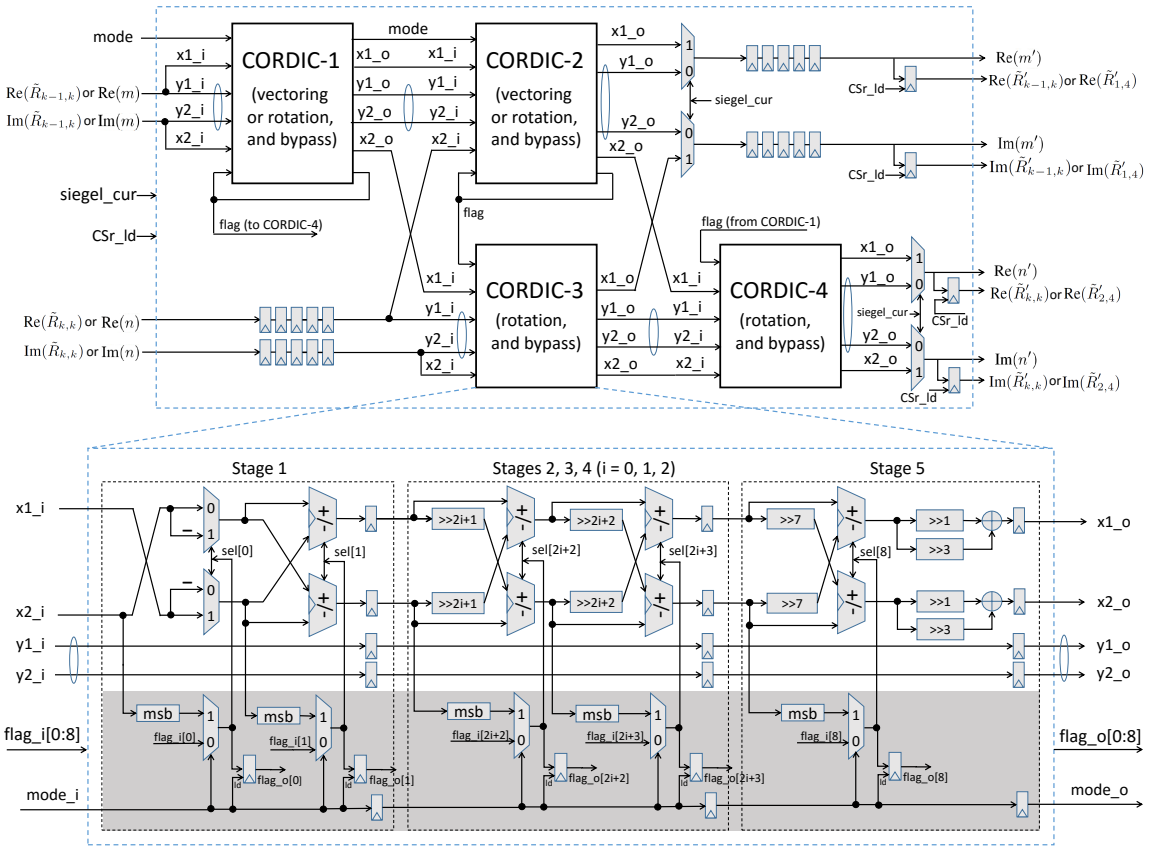


Figure 8.2: Architecture of the two-angle CGR module for column swap. The shaded area only exists in CORDIC-1/2 for vectoring mode.

(i.e., output = input) denoted by ellipses in Fig. 8.2, which is adopted to facilitate the data transfer between pipelined stages in case that the Siegel condition is satisfied (i.e., $siegel\_cur = 0$). To simplify the hardware design and timing schedule, ten DFFs are inserted before CORDIC-3 and after CORDIC-2, such that the input/output data streams are synchronized. The input/output notations follow Section 6.1.2, where the input $\tilde{R}_{k-1,k}$ and $\tilde{R}_{k,k}$ are used for vectoring mode while the input m and n indicate the other $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ or $\tilde{\boldsymbol{R}}$ entries for rotation mode. Note that the output data, $\tilde{R}'_{k-1,k}$ and $\tilde{R}'_{k,k}$ (i.e., the output of the vectoring mode of $\tilde{\boldsymbol{R}}$ in each fcLLL iteration as shown in Eq. (6.2)), or $\tilde{R}'_{1,4}$ and $\tilde{R}'_{2,4}$ (i.e., the last output of the rotation mode for $\tilde{\boldsymbol{R}}$ in the fcLLL iteration with $k = 2$), are registered with load input CSr_ld for the global timing optimization which will be discussed later in Section 8.1.3.

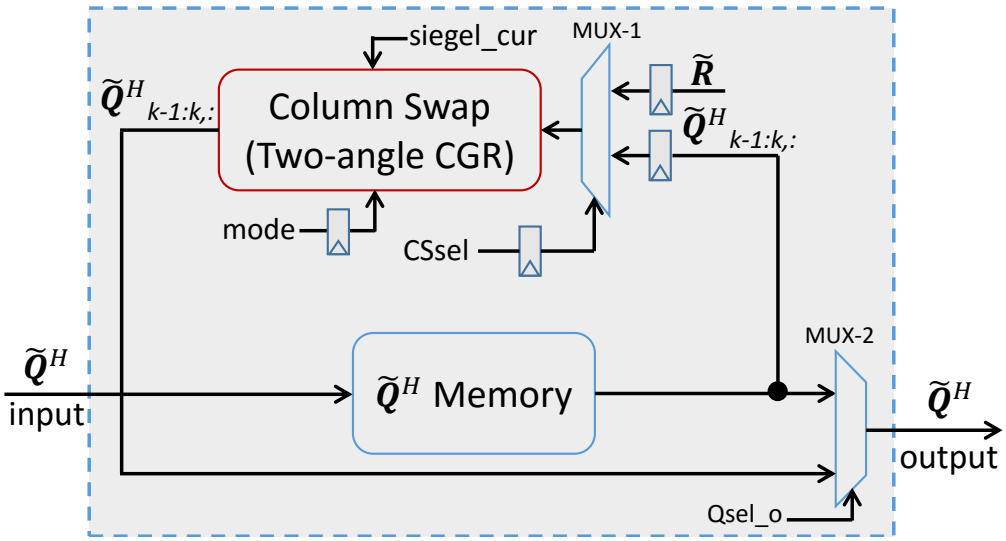


Figure 8.3: Modules cooperation for processing $\tilde{Q}^{\mathcal{H}}$ matrix.

8.1.2 Modules Cooperation for Processing Matrices

The detailed interconnection and cooperation among modules are ignored in the high-level architecture in Fig. 8.1 for simplification. In this section, we will elaborate these operations for $\tilde{Q}^{\mathcal{H}}$, $\tilde{R}$ and T matrices as follows, respectively.

The modules cooperation for processing $\tilde{Q}^{\mathcal{H}}$ matrix is illustrated in Fig. 8.3. The input data is stored in SRAM memory in each pipelined stage. Since the processing for $\tilde{Q}^{\mathcal{H}}$ matrix only exists in column swap as shown in Table 6.1, the $\tilde{Q}^{\mathcal{H}}$ memory output is only connected to the two-angle CGR module besides connected to the next stage. To save hardware cost, the two-angle CGR module is time-multiplexed between $\tilde{Q}^{\mathcal{H}}$ and $\tilde{R}$ matrices for column swap. Note that DFFs are inserted before the input of the two-angle CGR module, at the signals $CSsel$ and $mode$ (to trigger vectoring or rotation mode) from controller and at the $\tilde{Q}^{\mathcal{H}}$ and $\tilde{R}$ data from SRAM memories, to improve the clock frequency. Also note that the updated $\tilde{Q}^{\mathcal{H}}$ from the two-angle CGR module is directly connected to the output multiplexer for the next pipelined stage, which saves memory operations.

The processing of $\tilde{R}$ matrix involves all the operations of size reduction, Siegel

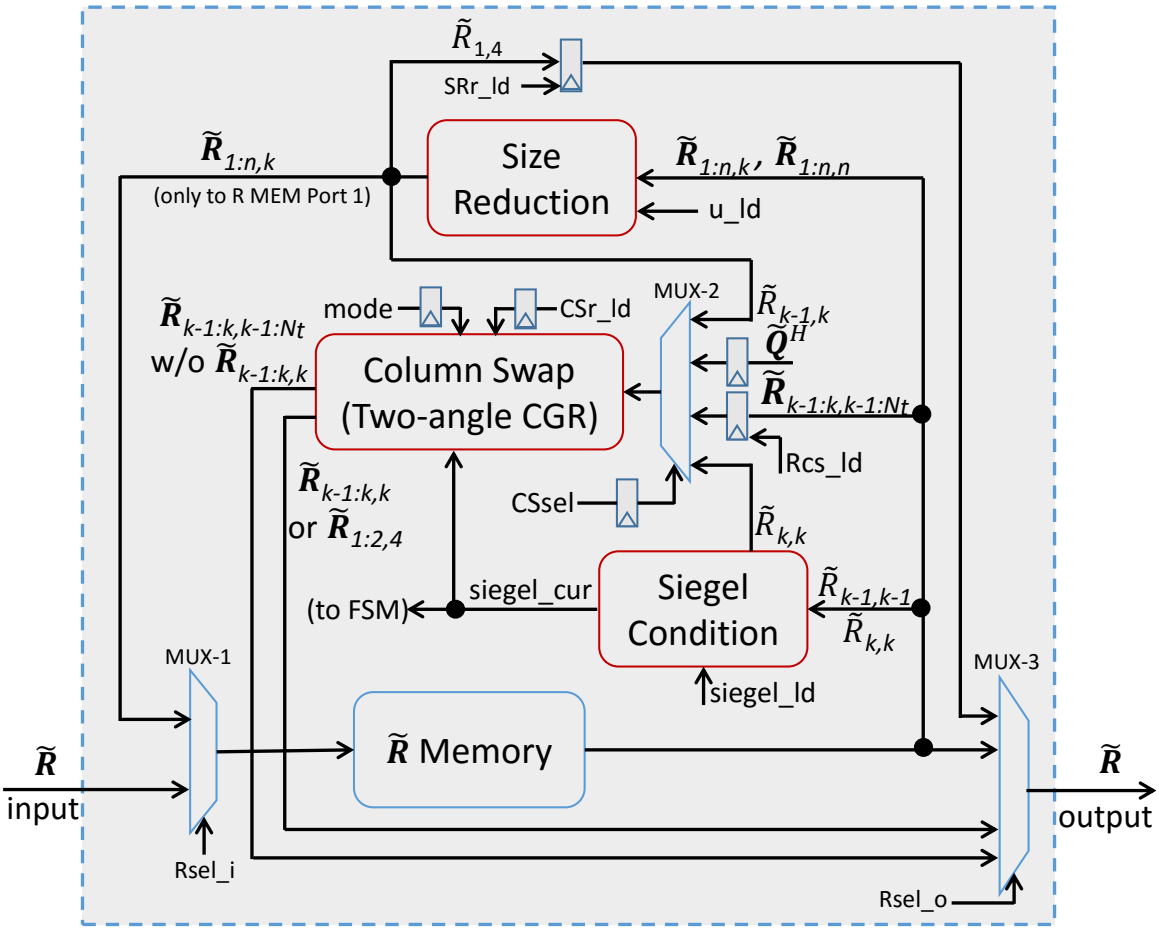


Figure 8.4: Modules cooperation for processing $\tilde{\boldsymbol{R}}$ matrix.

condition, and column swap as shown in Table 6.1. The detailed modules cooperation is illustrated in Fig. 8.4. At each pipelined stage, the size reduction and Siegel condition start to execute before the column swap by fetching the corresponding data from the $\tilde{\boldsymbol{R}}$ memory. Then, the updated $\tilde{R}_{k-1,k}$ from size reduction and the unchanged $\tilde{R}_{k,k}$ from Siegel condition (as shown in Fig. 7.3) are connected to the input multiplexer of the two-angle CGR module. The other $\tilde{\boldsymbol{R}}$ entries for the input of the two-angle CGR module are fetched directly from the $\tilde{\boldsymbol{R}}$ memory. The output of size reduction module is stored back to $\tilde{\boldsymbol{R}}$ memory, except the $\tilde{R}_{1,4}$ at the pipelined fcLLL stage with $k = 4$. The $\tilde{R}_{1,4}$ is registered and transferred to the next stage later to avoid memory conflict which will be discussed in Section 8.1.3. The output *siegel_cur* signal from Siegel condition module is connected to both the two-angle

CGR module (to decide whether performing column swap) and the controller module (to decide the values of control signals). Similarly as the the processing for $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ matrix, the output of $\tilde{\boldsymbol{R}}$ matrix from the two-angle CGR module is directly connected to the output multiplexer for the next pipelined stage. Note that the output for $\tilde{\boldsymbol{R}}$ matrix is divided into two parts which corresponds to the design in Fig. 8.2 for global timing optimization.

The modules cooperation for processing $\boldsymbol{T}$ matrix is illustrated in Fig. 8.5. As shown in Table 6.1, the operations for $\boldsymbol{T}$ matrix only exists in size reduction, so the $\boldsymbol{T}$ memory output is connected to the input of the size reduction module besides connected to the next stage. It is worthy to note that the number of $\boldsymbol{T}$ entries needed to be processed in size reduction is larger than that of $\tilde{\boldsymbol{R}}$ entries (see Lines 6-7 of Table 6.1). For example, compared to 12 times for $\tilde{\boldsymbol{R}}$ memory read/write operations in the fcLLL pipelined stage with $k = 4$, the $\boldsymbol{T}$ memory read/write operations would be 24 times (i.e., 24 processing cycles just for memory operations) if each each memory cell stores one $\boldsymbol{T}$ entry. This leads to sophisticated hardware design to achieve our final throughput with 26 processing cycles, if it is not impossible. Considering that the wordlength of $\boldsymbol{T}$ entries in fixed-point design is half that of the $\tilde{\boldsymbol{R}}$ entries, we adopt the design that each memory cell stores two $\boldsymbol{T}$ entries. Therefore, the number of $\boldsymbol{T}$ memory operations is reduced by half. Since each memory read/write operation produces two $\boldsymbol{T}$ entries, the input and output of the size reduction module for $\boldsymbol{T}$ matrix need to separate and concatenate the two entries, respectively. The corresponding functionality is implemented by two DFFs, one concatenation, and one multiplexer as shown in the Fig. 8.5. Note that the two entries of $\boldsymbol{T}$ memory $\{T_{1,4}, T_{2,4}\}$ and $\{T_{3,4}, T_{4,4}\}$, i.e., the final updated $4th$ column of $\boldsymbol{T}$ matrix in the fcLLL pipelined stage with $k = 4$, are registered and transferred to the next stage later to avoid memory conflict which will be discussed in Section 8.1.3.

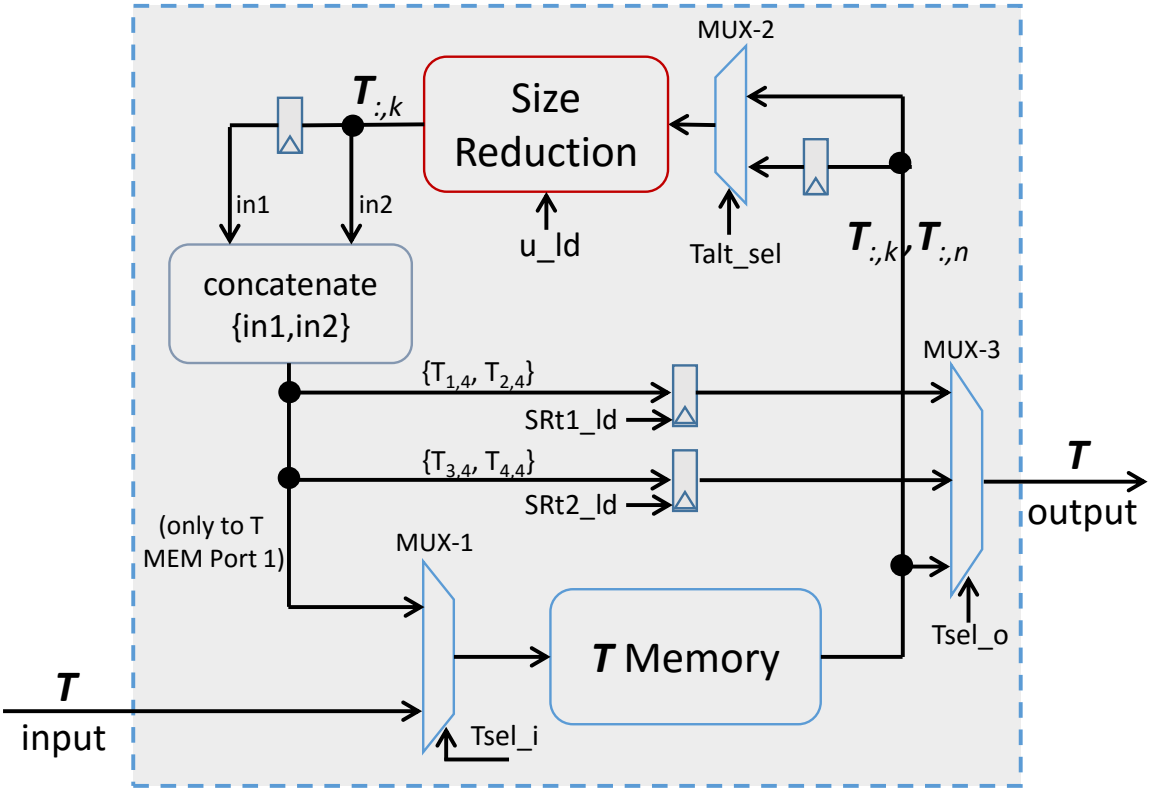


Figure 8.5: Modules cooperation for processing $\boldsymbol{T}$ matrix.

8.1.3 Timing Schedule and Data Transfer

Once the design of all datapaths is ready, careful timing schedule is needed to achieve high throughput and low latency. To estimate the minimum processing period of the whole pipelined architecture, let us consider the pipelined fcLLL stage with $k = 2$. Due to the data dependency, the updated $\tilde{R}_{1,2}$ from size reduction has to be obtained in order to do column swap for $\tilde{\boldsymbol{R}}$ and $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ matrices. The minimum number of cycles for the size reduction of $\tilde{R}_{1,2}$ is 4 (1 cycle for memory read plus 3 cycles for size reduction as show in Fig. 7.2); and the minimum number of cycles for the following column swap is 23 (15 cycles for the processing latency of the two-angle CGR module, plus 8 cycles for data input from the rows 1 and 2 of $\tilde{\boldsymbol{R}}$ and $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ matrices). This leads to minimum processing period of 27 cycles without considering the data transfer between pipelined stages with dual-port block RAMs. However, by

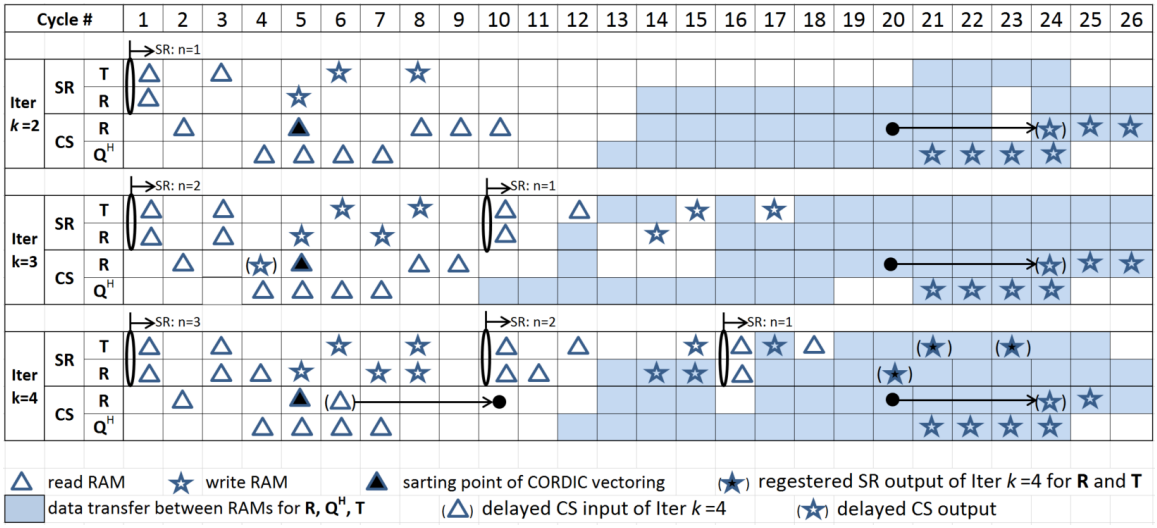


Figure 8.6: Timing schedule of the whole hardware architecture.

carefully designing the timing schedule and exploiting the "read before write" mode of dual-port block RAMs for data transfer, we can even achieve the throughput of processing period of 26 cycles as elaborated later. Note that for the timing schedule, one need to consider the bandwidth restriction of the dual-port memories. That is, during each cycle, one can fetch or store at most two complex numbers for $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ or $\tilde{\boldsymbol{R}}$ data, or four complex numbers for $\boldsymbol{T}$ data as indicated in Section 7.1.1.

The timing schedule in each pipelined fcLLL stage is implemented by a finite state machine (FSM) as a local controller. The detailed timing schedule of $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$, $\tilde{\boldsymbol{R}}$ and $\boldsymbol{T}$ matrices for each iteration index k is summarized in Fig. 8.6, where SR and CS denote size reduction and column swap operations, respectively. The operations of $\tilde{\boldsymbol{R}}$ in SR and CS are separated for better understanding, and the Siegel condition evaluation is included in CS for simplification. First, the size reduction operations for $\tilde{\boldsymbol{R}}_{1:n,k}$ and $\boldsymbol{T}_{:,k}$ are performed from Cycle 1, where the original data is fetched from memories and later the updated data is written back to memories. There are up to three turns of size reduction operations which corresponds to $n = 3$ (from Cycle 1), $n = 2$ (from Cycle 10), and $n = 1$ (from Cycle 16), in the fcLLL iteration with $k = 4$. Note that the third turn of size reduction starts at Cycle 16 which is before

the end of the second turn at Cycle 17 (the memory write for the updated $\boldsymbol{T}_{:,4}$). This design is chosen to archive the processing period of 26 cycles without memory read/write conflicts. Second, the Siegel condition evaluation is performed from Cycle 2 by fetching the two entries $\tilde{R}_{k-1,k-1}$ and $\tilde{R}_{k,k}$ from memory. Then, at the beginning of Cycle 5, we obtain the result of Siegel condition as well as the two entries $\tilde{R}_{k-1,k}$ (from size reduction module) and $\tilde{R}_{k,k}$ (from Siegel condition module), which are used for the first vectoring operations in two-angle CGR. Thus, the two-angle CGR for column swap starts at Cycle 5. Note that after the vectoring operations for $[\tilde{R}_{k-1,k}, \tilde{R}_{k,k}]^T$, we begin the rotation mode for the four vectors of $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ from Cycle 6 to Cycle 9 by fetching memory data from Cycle 4 to cycle 7, followed by the rotation mode for the remaining vectors of $\tilde{\boldsymbol{R}}$ at Cycle 10 by fetching memory data at Cycle 8. This design choice comes from two considerations. One is to avoid the $\tilde{\boldsymbol{R}}$ memory conflicts between size reduction and column swap operations; the other is that the we inserted one DFF/rigister before the input data from memory to the two-angle CGR module for frequency optimization as discussed in Section 8.1.2. The output cycles of the two-angle CGR for column swap are from Cycle 21 to Cycle 26, where the output data is directly transferred to the next pipelined stage instead of storing back to memory to improve throughput. Note that the first output at Cycle 20 is delayed to Cycle 24, such that the updated rows $k-1$ and k of $\tilde{\boldsymbol{R}}$ from the two-angle CGR are transferred to the RAM $\tilde{\boldsymbol{R}}$ of next iteration continually, which can facilitate the data transfer between pipelined stages. To avoid memory conflicts, the outputs of the size reduction in the final turn for $\tilde{\boldsymbol{T}}$ and $\tilde{\boldsymbol{R}}$ at iteration $k = 4$ are registered instead of writing back to memories, and these registered data are later transferred to the next stage.

The detailed data transfer between pipelined fcLLL stages is summarized in Fig. 8.7, Fig. 8.8, and Fig. 8.9, for $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$, $\tilde{\boldsymbol{R}}$ and $\boldsymbol{T}$ matrices, respectively. The property of the "read before write" mode of the dual-port block RAM is utilized to save cycles

for data transfer, such that the data transfer between two pipelined stages only exists one cycle delay. Take the data transfer of row-1 and row-2 matrix data between the iterations with $k = 4$ and $k = 3$ in Fig. 8.7 for example, we can input new data at Cycles 14-17 in the iteration with $k = 4$, while the old data in the iteration with $k = 4$ can be transferred to the iteration with $k = 3$ at Cycles 15-18. Another way to save cycles for data transfer is to output the results of column swap directly to the next stage instead of writing back to memories, which is shown in Fig. 8.7 and Fig. 8.8 for $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ and $\tilde{\boldsymbol{R}}$ matrices, respectively. A third way to save cycles for data transfer is to implement the column exchanging operations (see Line 13 in Table 6.1) during the data transfer between pipelined stages. For $\tilde{\boldsymbol{R}}$ data transfer, this is achieved by generating the corresponding addresses based on the Siegel condition signal *siegel_pre* as shown in Fig. 8.8; while for $\boldsymbol{T}$ data transfer, this is achieved by generating the corresponding control signal for the multiplexers based on the Siegel condition signal *siegel_cur* as shown in Fig. 8.9. The occupied cycles for data transfer correspond to the shaded area in Fig. 8.6, without memory conflicts with other modules. Therefore, by adopting the timing schedule in Fig. 8.6 for the pipelined architecture in Fig. 8.1, we achieve the throughput of processing period of 26 cycles per matrix.

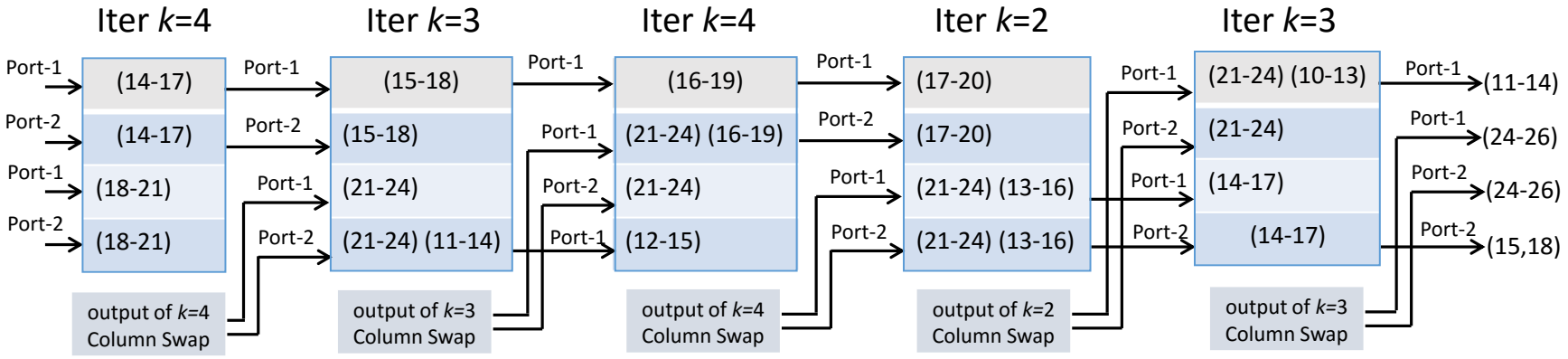


Figure 8.7: Data transfer between pipelined fcLLL stages for $\tilde{\boldsymbol{Q}}^{\mathcal{H}}$ matrix.

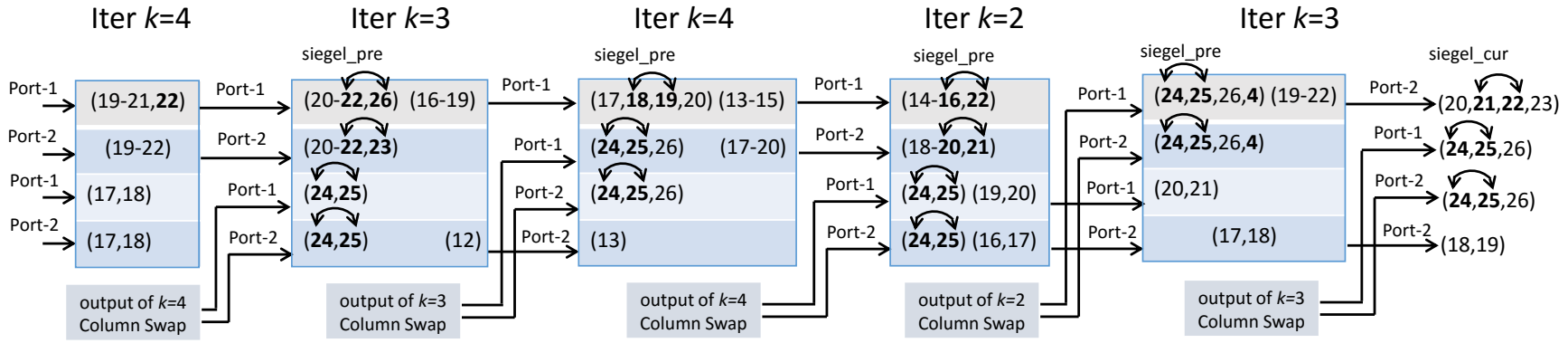


Figure 8.8: Data transfer between pipelined fcLLL stages for $\tilde{\boldsymbol{R}}$ matrix.

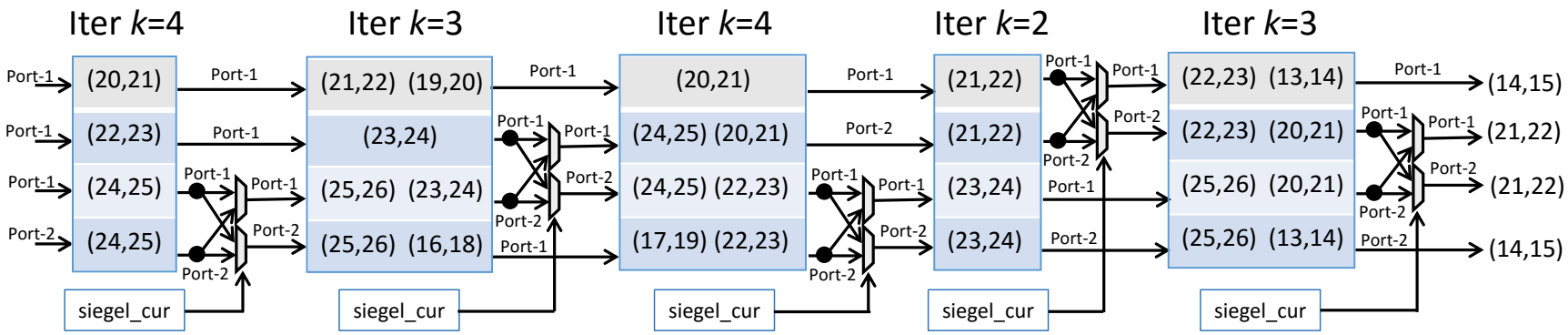


Figure 8.9: Data transfer between pipelined fcLLL stages for $\boldsymbol{T}^T$ matrix.

8.2 Implementation and Comparison

Fig. 8.10 compares the uncoded BER results of different fcLLL algorithms (i.e., Sequential fcLLL, Even-odd fcLLL, Incremental fcLLL, and the modified Incremental fcLLL) as well as the original LLL algorithm in the complex LR-aided MMSE K-best detector [79] for a 4×4 MIMO system with 64-QAM, where sorted QR is adopted for all LRs to reduce the number of iterations [84] and $K = 3$ candidates are used in the K-best detector. We set fixed 5 iterations for LLL and all fcLLL algorithms in simulation. First, we compare the floating-point results (solid lines). It can be seen that original and modified Incremental fcLLLs achieve almost the same performance as the ML detector, and exhibit about 0.5 dB, 2.4 dB, and 6.8 dB performance gains at BER=10^{-4} compared to Even-odd fcLLL, Sequential fcLLL, and original LLL, respectively. Although we only consider 4×4 MIMO system, the Incremental fcLLL algorithm generates more performance gain in higher dimension MIMO systems as shown in[72]. Next, the fixed-point performance of the proposed modified Incremental fcLLL is depicted in dashed line, and the performance loss is only about 0.27 dB at BER=10^{-4} compared to the floating-point version.

8.2.1 Implementation Results

To evaluate the proposed pipelined architecture, we implemented it in Verilog under the FPGA RTL design flow, which includes Synopsys Synplify 2012 for synthesis, Xilinx ISE 14.7 for place-and-rout (PAR) with the synthesis results, and Xilinx ISim for verification compared with the C++ fixed-point model in Section 6.2. The implementation results on Xilinx Virtex-4/5/7 FPGAs are summarized in the last column of Table 8.2 and Table 8.3. It can be seen that our design brings a processing throughput up to 9.92 million matrices per second. The distribution of the main hardware resource (FPGA Slices) for each module is also provided for reference as shown in Table 8.1. It is clear that the two-angle CGR module for column swap occupies most

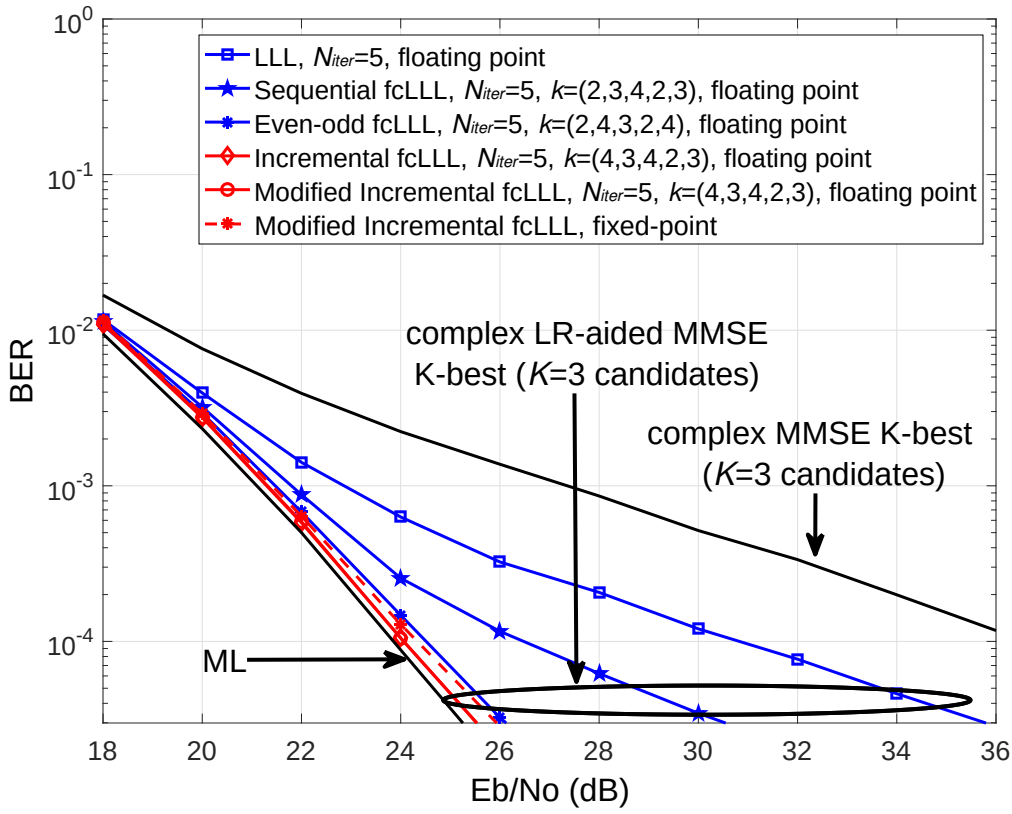


Figure 8.10: BER comparison of different LR algorithms with fixed $N_{iter} = 5$ iterations in the complex LR-aided MMSE K-best detector ($K = 3$ candidates) for a 4×4 MIMO system with 64-QAM.

of resources, which is the reason that we optimize the two-angle CGR module before other modules in the fixed-point design in Section 6.2.

Table 8.1: Distribution of FPGA Slices for Different Module

Size Reduction	21%
Siegel Condition	4%
Column Swap (2-angle CGR)	70%
Controller (FSM)	5%

8.2.2 Comparison with FPGA Implementations

Table 8.2 summarizes the recent FPGA implementations of LLL variants for 4×4 MIMO systems, based on the Xilinx FPGAs. Since all the implementations have similar Virtex-4/5 FPGAs (except [5] with Virtex-2 Pro FPGA) and *Throughput* =

Clock Frequency ÷ (Cycles/Matrix), we mainly focus on the throughput and FPGA resources for discussion. First, compared to the implementation in [19] and [18] with the same FPGA type, our design has the advantage of fixed throughput, higher clock frequency, and fewer cycles per matrix at the cost of around two-fold FPGA slices. In terms of average and worst throughput, our implementation exhibits near two-fold and seventeen-fold improvements compared to [18], respectively. Second, compared to the implementation in [5] with 24 multipliers, our implementation does not need multiplier and has much higher throughput. Third, compared to the RS-LLL with the same Virtex-4 [9], our design not only has the advantage of fixed throughput, but also has higher throughput than the average throughput of [9]. Fourth, even though the design in [30] requires only 12 cycles to process one matrix, it has pretty low clock frequency. Therefore our design still demonstrates about eleven-fold improvement in throughput compared to [30].

8.2.3 Comparison with ASIC and ASIP Implementations

Table 8.3 summarizes the recent ASIC/ASIP implementations of LLL variants for 4×4 MIMO systems, which are compared to our implementation on Xilinx Virtex-7 FPGA. First, our design has the advantage of fixed throughput while not all ASIC/ASIP designs possess this property. The fixed throughput in our design is higher than other existing ASIC/ASIP realizations with fixed throughput. Second, our design has similar pipelined structure as that of [55]. Since our design is based on Incremental fcLLL instead of Sequential fcLLL as that of [55], the required number of iterations is reduced from 9 as in [55] to 5 in our design. By incorporating the proposed two-angle CGR and careful timing schedule, our design achieves 26 cycles per matrix processing. Therefore, even though our frequency (based on FPGA) is much lower than that of [55] (based on ASIC), our final throughput is still higher

than the result of [55]. Compared to the ASIP realization in [56], our design offers advantages in clock frequency, cycles per matrix, and throughput (near nine-fold improvement in throughput). For the ASIC version of [30], although it has over four-fold improvement in frequency compared to its own FPGA version, our design still has much higher clock frequency which leads to over three-fold improvement in throughput compared to [30].

Table 8.2: Comparison of FPGA Implementations of the LLL Variants for 4×4 MIMO Systems.

Reference		ICCSC'2008 [19]	ICC'2009 [5]	ISCAS'2010 [9]	TCAS-I'2011 [18]	JCN'2011 [69]	TCAS-II'2011 [30]	This work	
LR Algorithm		LLL	LLL	RS-LLL	LLL	FSR-LLL	Even-odd fcLLL	Pipelined Incremental fcLLL	
FPGA Platform Device Number		Virtex-5 XC5VLX110-3	Virtex-2 Pro XC2VP30-7	Virtex-4 XC4VLX160-12	Virtex-5 XC5VLX110-3	Virtex-5 XC5VFX130T-n.a.	Virtex-4 XC4VLX60-n.a.	Virtex-4 XC4VLX160-12	Virtex-5 XC5VLX110-3
FPGA Resources	Slices	1,712	7,349	4,805	1,758	2,335	11,330	7,627	3,728
	Block RAMs	n.a.	69	0	n.a.	n.a.	n.a.	15	15
	Multipliers	10	24	18	4	n.a.	n.a.	0	0
Clock Frequency (MHz)		163	100	79	206	155	8.7	161	220
Cycles/Matrix		130^{avg}, $\text{n.a.}^{\text{worst}}$	420^{avg}, $\text{n.a.}^{\text{worst}}$	14^{avg}, $\text{n.a.}^{\text{worst}}$	49^{avg}, 447^{worst}	84^{fixed}, $\text{n.a.}^{\text{worst}}$	12^{fixed}	26^{fixed}	26^{fixed}
Throughput (MMat/s)		1.25^{avg}, $\text{n.a.}^{\text{worst}}$	0.24^{avg}, $\text{n.a.}^{\text{worst}}$	5.6^{avg}, $\text{n.a.}^{\text{worst}}$	4.20^{avg}, 0.46^{worst}	1.85^{avg}, $\text{n.a.}^{\text{worst}}$	0.73^{fixed}	6.19^{fixed}	8.46^{fixed}

Note: n.a. indicates "not available"; avg indicates "average".

Table 8.3: Comparison of ASIC/ASIP Implementations of the LLL Variants for 4×4 MIMO Systems.

Reference		ICCSC'2008 [82]	ISCAS'2010 [9]	TCAS-II'2011 [30]	TVLSI'2013 [55]	TSP'2013 [2]	TCAS-I'2014 [52]	ISCAS'2015 [56]	This work
LR Algorithm		SA	RS-LLL	Even-odd fcLLL	Sequential fcLLL	Sequential fcLLL	SA	LLL	Pipelined Incremental fcLLL
Hardware Platform		65nm ASIC	130nm ASIC	90nm ASIC	130nm ASIC	40nm ASIP	90nm ASIC	90nm ASIP	Virtex-7[†]
Hardware Resources	Gates Equivalent	67 kGE	107 kGE	200 kGE	125 kGE	6,364 kGE	112 kGE	405 kGE	-
	Slices	-	-	-	-	-	-	-	3,723
	Block RAMs	-	-	-	-	-	-	-	15
Clock Frequency (MHz)		400	333	37	352	700	360	210	258
Cycles/Matrix		n.a.^{avg}, 1368^{worst}	14^{avg}, $\text{n.a.}^{\text{worst}}$	12^{fixed}	40^{fixed}	21^{avg}, $\text{n.a.}^{\text{worst}}$	11.25^{avg}, 24^{worst}	187^{fixed}	26^{fixed}
Throughput (MMat/s)		n.a.^{avg}, 0.29^{worst}	23.8^{avg}, $\text{n.a.}^{\text{worst}}$	3.08^{fixed}	8.80^{fixed}	33.33^{avg}, $\text{n.a.}^{\text{worst}}$	32^{avg}, 15^{worst}	1.12^{fixed}	9.92^{fixed}

[†]Device number of the Xilinx Virtex-7 FPGA: XC7VX485T-3

8.3 Conclusion

In this chapter, we develop a pipelined hardware architecture based on the modified Incremental fcLLL algorithm. To reduce complexity and latency, we improve the two-angle CGR scheme for column swap, fully utilize the properties of the dual-port memories for data read/write and transfer, and optimize the timing schedule such that the final architecture on FPGA can produce a matrix every 26 cycles, resulting in much higher throughput than state-of-the-art FPGA implementations, and similar even better throughput than the recent ASIC/ASIP implementations.

CHAPTER IX

CONCLUDING REMARKS

9.1 Contributions

The objective of the proposed research is to design low-complexity and high-performance enhanced LLL algorithms for LR-aided MIMO detectors, in both theory and hardware implementation. The primary contributions of this dissertation are listed below:

- Analyzed the relationship between the error performance of LR-aided MIMO detectors and the LLL algorithms;
- Proposed two enhanced greedy LLL algorithms with much faster convergence and lower complexity than the existing greedy LLL algorithms;
- Proposed Incremental fcLLL algorithms with much faster convergence and lower complexity than the existing fcLLL algorithms;
- Examined the proposed greedy LLL and Incremental fcLLL algorithms by extensive simulations and demonstrated their advantages compared to existing solutions;
- Modified the proposed Incremental fcLLL algorithm by eliminating all computationally intensive operations for efficient hardware implementation;
- Developed a fixed-point conversion methodology for the modified Incremental fcLLL algorithm and completed the corresponding fixed-point design;
- Developed a low-complexity iterative hardware architecture to implement the modified Incremental fcLLL algorithm;

- Developed a high-throughput pipelined hardware architecture to implement the modified Incremental fcLLL algorithm;

- Implemented both iterative and pipelined architectures of the modified Incremental fcLLL algorithm by Verilog on Xilinx Virtex-4/5/7 FPGA devices and demonstrated their advantages compared to existing hardware implementations.

9.2 Suggestions for Future Research

The following is a list of interesting research topics that can be pursued as extensions of this dissertation:

- The proposed enhanced greedy LLL and fcLLL algorithms and implementations can be applied to the soft-output MIMO detectors as in [4, 49, 58, 94], as well as the MIMO precoding techniques as in [12, 81, 88, 105].

- The proposed enhanced greedy LLL and fcLLL algorithms and implementations can be investigated in the new area of combining MIMO techniques with spatial modulation, visible light communication, or mm-Wave systems [16, 68, 89].

- Since the main computational part of both QR and fcLLL can be realized by Govens rotation operations, the QR preprocessing part can be implemented together with the proposed fcLLL architecture to improve the hardware's efficiency.

- The developed iterative and pipelined architectures of the fcLLL algorithm can be extended to more complex MIMO systems with higher dimension, e.g., 8×8 and 16×16 systems. This extension can be realized by reusing the current datapath but the timing schedule needs to be redesigned for best-achievable throughput.

9.3 Publications

The following is the list of publications related to this dissertation.

Journal Publications (published/submitted/to be submitted)

J1. **Q. Wen** and X. Ma, "Efficient greedy LLL algorithms for lattice decoding," *IEEE Trans. Wireless Commun.*, vol. 15, no. 5, pp. 3560-3572, May 2016.

J2. **Q. Wen** and X. Ma, "High throughput implementation of incremental fixed-complexity LLL algorithm," to be submitted to *IEEE Trans. Circuits Syst. I.*, 2017.

J3. **Q. Wen** and X. Ma, "Low-complexity iterative implementation of incremental fixed-complexity LLL algorithm for MIMO detection," to be submitted to *IEEE Trans. Circuits Syst. II.*, 2017.

J4. **Q. Wen**, X. Ma, Z. Yu, and G. Zhou, "Optimizing nonlinear effects for multi-user OFDM-Based digital vertical beamforming transmissions," submitted to *IEEE Trans. Broadcast.*, 2017.

J5. **Q. Wen** and X. Ma, "Generalized fixed-Structure LLL and ELLL algorithms for low-complexity high-speed MIMO detection," to be submitted to *IEEE Wireless Commun. Letters*, 2017.

Conference Publications (published)

C1. **Q. Wen** and X. Ma, "VLSI implementation of incremental fixed-complexity LLL lattice reduction for MIMO detection," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, Montreal, Canada, pp. 1898-1901, May 2016.

C2. **Q. Wen** and X. Ma, "Fixed-complexity variants of the effective LLL algorithm with greedy convergence for MIMO detection," in *Proc. IEEE 41th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Shanghai, China, pp. 3826-3830, Mar. 2016.

C3. **Q. Wen**, Q. Zhou, and X. Ma, "An enhanced fixed-complexity LLL algorithm for MIMO detection," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, Austin, TX, pp. 3231-3236, Dec. 2014.

C4. **Q. Wen** and X. Ma, "An Efficient Greedy LLL Algorithm for MIMO Detection," in *Proc. IEEE 33th Military Communications Conference (MILCOM)*, Baltimore, MD, pp. 550-555, Oct. 2014.

C5. **Q. Wen**, Q. Zhou, C. Zhao, and X. Ma, "Fixed-point realization of lattice-reduction aided MIMO receivers with complex K-best algorithm," in *Proc. IEEE 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, pp. 5031-5035, May 2013.

C6. **Q. Wen**, S. Lee, and X. Ma, "Clipping effect on radiation pattern in downtilt beamforming," in *Proc. IEEE 46th Asilomar Conference on Signals, Systems, and Computers (ASILOMAR)*, Pacific Grove, CA, pp. 1873-1877, Nov. 4-7, 2012.

C7. S. Lee, X. Ma, and **Q. Wen**, "Transmitter-side timing adjustment to mitigate interference between multiple nodes for OFDMA mesh network," in *Proc. IEEE 46th Asilomar Conference on Signals, Systems, and Computers (ASILOMAR)*, Pacific Grove, CA, pp. 1957-1961, Nov. 4-7, 2012.

Patents

P1. X. Ma and **Q. Wen**, "Multi-input multi-output (mimo) detection systems," in *US Patent* No. 15/056,986, Publication date Sep 1, 2016.

P2. **Q. Wen** and X. Ma, "High efficient wireless detectors with incremental fixed-complexity LLL algorithms," disclosure submitted on Nov. 26, 2014.

APPENDIX A

PROOF OF PROPOSITION 2.1

Let us consider the effect of size reduction of each column on the results of LR-aided K-best detectors. When the size reduction proceeds to column c ($2 \leq c \leq N_t$), let $\tilde{\boldsymbol{R}}$, $\boldsymbol{T}$ and $\tilde{\boldsymbol{R}}'$, $\boldsymbol{T}'$ represent the values before and after the size reduction, respectively. Then, the operations of the size reduction on column c (Lines 4-10 of Table 2.1) can be rewritten as

$$\tilde{\boldsymbol{R}}'_{:,c} = \tilde{\boldsymbol{R}}_{:,c} - \sum_{n=1}^{c-1} u_{n,c}\tilde{\boldsymbol{R}}_{:,n}, \tag{A.1a}$$

$$\boldsymbol{T}'_{:,c} = \boldsymbol{T}_{:,c} - \sum_{n=1}^{c-1} u_{n,c}\boldsymbol{T}_{:,n}, \tag{A.1b}$$

where $u_{n,c} = \lfloor R_{n,c}/R_{n,n} \rceil$. Since only the upper-off-diagonal elements of the cth column of the $\tilde{\boldsymbol{R}}$ are updated as shown in (A.1a), the K partial candidates from the LR-aided K-best detector for the system (2.17) will be unchanged from N_tth to cth levels. Denote the partial candidates of the cth level in z domain as $\{\hat{\boldsymbol{z}}_k^{(c)}\}_{k=1}^K$ and the corresponding cost as $\{cost_k^{(c)}\}_{k=1}^K$, where $\hat{\boldsymbol{z}}_k^{(c)} = [\hat{z}_{k,c}^{(c)}, \hat{z}_{k,c+1}^{(c)}, \ldots, \hat{z}_{k,N_t}^{(c)}]^T$. When the detector proceeds to the $(c+1)$th layer, if the $\tilde{\boldsymbol{R}}$ is unchanged, the LR-aided K-best algorithm calculates the first child of each K existing parents nodes in z domain as

$$\begin{aligned}\hat{z}_{k,1\text{st}}^{(c-1)} &= \arg \min_{z_{k,1\text{st}}^{(c-1)} \in \mathbb{Z}_j} \left| \tilde{y}_{c-1} - \sum_{n=c}^{N_t} \tilde{R}_{c-1,n}\hat{z}_{k,n}^{(c)} - \tilde{R}_{c-1,c-1}z_{k,1\text{st}}^{(c-1)} \right|^2 \\ &= \arg \min_{z_{k,1\text{st}}^{(c-1)} \in \mathbb{Z}_j} \left| \frac{\tilde{y}_{c-1} - \sum_{n=c}^{N_t} \tilde{R}_{c-1,n}\hat{z}_{k,n}^{(c)}}{\tilde{R}_{c-1,c-1}} - z_{k,1\text{st}}^{(c-1)} \right|^2 = \left\lceil \frac{\tilde{y}_{c-1} - \sum_{n=c}^{N_t} \tilde{R}_{c-1,n}\hat{z}_{k,n}^{(c)}}{\tilde{R}_{c-1,c-1}} \right\rfloor,\end{aligned} \tag{A.2}$$

and the corresponding cost increment is

$$\triangle cost_{k,1\text{st}}^{(c-1)} = \left| \tilde{y}_{c-1} - \sum_{n=c}^{N_t} \tilde{R}_{c-1,n}\hat{z}_{k,n}^{(c)} - \tilde{R}_{c-1,c-1}\hat{z}_{k,1\text{st}}^{(c-1)} \right|^2. \tag{A.3}$$

By applying the substitution of $\tilde{\boldsymbol{R}}'$ from size reduction in (A.1a), the updated first child of the each K existing parents nodes is

$$\begin{aligned}\hat{z}_{k,1\mathrm{st}}^{(c-1)\prime} &= \arg\min_{z_{k,1\mathrm{st}}^{(c-1)\prime}\in\mathbb{Z}_j}\left|\tilde{y}_{c-1}-\sum_{n=c}^{N_t}\tilde{R}'_{c-1,n}\hat{z}_{k,n}^{(c)}-\tilde{R}'_{c-1,c-1}z_{k,1\mathrm{st}}^{(c-1)\prime}\right|^2 \\ &= \arg\min_{z_{k,1\mathrm{st}}^{(c-1)\prime}\in\mathbb{Z}_j}\left|\tilde{y}_{c-1}-\sum_{n=c}^{N_t}\tilde{R}_{c-1,n}\hat{z}_{k,n}^{(c)}+u_{c-1,c}\tilde{R}_{c-1,c-1}\hat{z}_{k,c}^{(c)}-\tilde{R}_{c-1,c-1}z_{k,1\mathrm{st}}^{(c-1)\prime}\right|^2 \\ &= \left\lceil\frac{\tilde{y}_{c-1}-\sum_{n=c}^{N_t}\tilde{R}_{c-1,n}\hat{z}_{k,n}^{(c)}}{\tilde{R}_{c-1,c-1}}+u_{c-1,c}\hat{z}_{k,c}^{(c)}\right\rfloor = \hat{z}_{k,1\mathrm{st}}^{(c-1)}+u_{c-1,c}\hat{z}_{k,c}^{(c)}, \end{aligned} \tag{A.4}$$

and the corresponding updated cost increment is

$$\begin{aligned}{}_{\triangle}cost_{k,1\mathrm{st}}^{(c-1)\prime} &= \left|\tilde{y}_{c-1}-\sum_{n=c}^{N_t}\tilde{R}'_{c-1,n}\hat{z}_{k,n}^{(c)}-\tilde{R}'_{c-1,c-1}\hat{z}_{k,1\mathrm{st}}^{(c-1)\prime}\right|^2 \\ &= \left|\tilde{y}_{c-1}-\sum_{n=c}^{N_t}\tilde{R}_{c-1,n}\hat{z}_{k,n}^{(c)}+u_{c-1,c}\tilde{R}_{c-1,c-1}\hat{z}_{k,c}^{(c)}-\tilde{R}_{c-1,c-1}(\hat{z}_{k,1\mathrm{st}}^{(c-1)}+u_{c-1,c}\hat{z}_{k,c}^{(c)})\right|^2 \\ &= \left|\tilde{y}_{c-1}-\sum_{n=c}^{N_t}\tilde{R}_{c-1,n}\hat{z}_{k,n}^{(c)}-\tilde{R}_{c-1,c-1}\hat{z}_{k,1\mathrm{st}}^{(c-1)}\right|^2 = {}_{\triangle}cost_{k,1\mathrm{st}}^{(c-1)}. \end{aligned} \tag{A.5}$$

It can be seen that the updated first child in (A.4) is just a shift of $u_{c-1,c}\hat{z}_{k,c}^{(c)}$ (i.e., $u_{c-1,c}$ times its parent node) in the two dimensional plane with Gaussian integers. And the corresponding cost increment of each first child keeps unchanged as shown in (A.5). Next, based on the first child, the LR-aided K-best algorithm can find the next child by employing the complex Schnorr-Euchner (SE) strategy (more detailed realization can be found in Table I of [79]), and the mth ($1 < m \leq \mathcal{M}$) children of each K existing parents nodes with and without size reduction can be obtained as

$$\hat{z}_{k,m\mathrm{th}}^{(c-1)} = \hat{z}_{k,1\mathrm{st}}^{(c-1)} + {}_{\triangle}z_{m\mathrm{th}}, \tag{A.6a}$$

$$\hat{z}_{k,m\mathrm{th}}^{(c-1)\prime} = \hat{z}_{k,m\mathrm{th}}^{(c-1)} + u_{c-1,c}\hat{z}_{k,c}^{(c)}, \tag{A.6b}$$

where ${}_{\triangle}z_{m\mathrm{th}} \in \mathbb{Z}_j$ denotes the shift from the first child generated by the complex SE strategy. Since $\hat{z}_{k,m\mathrm{th}}^{(c-1)\prime}$ has the same shift as in (A.4), we can derive ${}_{\triangle}cost_{k,m\mathrm{th}}^{(c-1)\prime} = {}_{\triangle}cost_{k,m\mathrm{th}}^{(c-1)}$ similarly like (A.5). Now, we can conclude that, with the full size reduction,

all the expanded children of each parent node have the same shift and their corresponding cost increments are unchanged, so the updated K best partial candidates at the $(c-1)$th level can be obtained

$$\hat{z}_{k,n}^{(c-1)\prime} = \begin{cases} \hat{z}_{k,n}^{(c-1)}, & c \le n \le N_t \\ \hat{z}_{k,c-1}^{(c-1)} + u_{c-1,c}\hat{z}_{k,c}^{(c-1)}, & n = c-1. \end{cases} \tag{A.7}$$

Note that the notation in (A.7) is $u_{c-1,c}\hat{z}_{k,c}^{(c-1)}$ instead of $u_{c-1,c}\hat{z}_{k,c}^{(c)}$, since the shift of each node in the $(c-1)$th layer should be $u_{c-1,c}$ times its parent node. Based on the aforementioned process, it is straightforward to use induction to prove that the updated K best candidates at the first level satisfy

$$\hat{z}_{k,n}^{(1)\prime} = \begin{cases} \hat{z}_{k,n}^{(1)}, & c \le n \le N_t \\ \hat{z}_{k,n}^{(1)} + u_{n,c}\hat{z}_{k,c}^{(1)}, & 1 \le n \le c-1. \end{cases} \tag{A.8}$$

Then, based on (A.8) and (A.1b), we have

$$\begin{aligned} \boldsymbol{T}'\hat{\boldsymbol{z}}_k^{(1)\prime} &= \sum_{n=1}^{N_t} \boldsymbol{T}'_{:,n}\hat{z}_{k,n}^{(1)\prime} \\ &= \sum_{n=1}^{c-1} \boldsymbol{T}_{:,n}(\hat{z}_{k,n}^{(1)} + u_{n,c}\hat{z}_{k,c}^{(1)}) + (\boldsymbol{T}_{:,c} - \sum_{n=1}^{c-1} u_{n,c}\boldsymbol{T}_{:,n})\hat{z}_{k,n}^{(1)} + \sum_{n=c+1}^{N_t} \boldsymbol{T}_{:,n}\hat{z}_{k,n}^{(1)} \\ &= \sum_{n=1}^{N_t} \boldsymbol{T}_{:,n}\hat{z}_{k,n}^{(1)} = \boldsymbol{T}\hat{\boldsymbol{z}}_k^{(1)}. \end{aligned} \tag{A.9}$$

Therefore, the final updated output of the LR-aided K-best detector in the $\boldsymbol{s}$ domain is

$$\begin{aligned} \hat{\boldsymbol{s}}' &= \arg \min_{\hat{\boldsymbol{s}}'_k = \mathcal{Q}[2\boldsymbol{T}'\hat{\boldsymbol{z}}_k^{(1)\prime} - \mathbf{1}(1+j)]} \|\boldsymbol{y} - \boldsymbol{H}\hat{\boldsymbol{s}}'_k\|^2 \\ &= \arg \min_{\hat{\boldsymbol{s}}_k = \mathcal{Q}[2\boldsymbol{T}\hat{\boldsymbol{z}}_k^{(1)} - \mathbf{1}(1+j)]} \|\boldsymbol{y} - \boldsymbol{H}\hat{\boldsymbol{s}}_k\|^2 = \hat{\boldsymbol{s}}, \end{aligned} \tag{A.10}$$

which shows that the result is the same as the case without size reduction. So the size reduction does not affect the results of LR-aided K-best detection. We can conclude that the ELLL and LLL have the same error performance in LR-aided K-best detection.

APPENDIX B

PROOF OF PROPOSITION 3.1

Let us consider the $\tilde{\boldsymbol{R}}$ matrix of the primal basis here, but the following proof also applies to the $\tilde{\boldsymbol{R}}^{\star}$ matrix of the dual basis. Suppose that a column swap (Lines 12-15 of Table 2.1) happens at column pair $(k-1, k)$. Let $\tilde{R}_{k-1,k-1}, \tilde{R}_{k-1,k}, \tilde{R}_{k,k}$ denote the values before the column swap and $\tilde{R}'_{k-1,k-1}, \tilde{R}'_{k-1,k}, \tilde{R}'_{k,k}$ denote the updated values after the column swap. Since the Lovász condition is not held before the column swap, we have

$$|\tilde{R}_{k,k}|^2 + |\tilde{R}_{k-1,k}|^2 < \delta |\tilde{R}_{k-1,k-1}|^2 \leq |\tilde{R}_{k-1,k-1}|^2, \tag{B.1}$$

where the second inequality comes from $1/2 < \delta \leq 1$. Since $\boldsymbol{\Theta}$ is a 2×2 unitary matrix, $||\boldsymbol{r}||^2 = ||\boldsymbol{\Theta}\boldsymbol{r}||^2$ holds for any 2×1 vector $\boldsymbol{r}$. Thus, if we take both columns $k-1$ and k into consideration, the updated $\tilde{R}'_{k-1,k-1}$, $\tilde{R}'_{k-1,k}$, and $\tilde{R}'_{k,k}$ satisfy

$$|\tilde{R}'_{k-1,k-1}|^2 + |\tilde{R}'_{k-1,k}|^2 + |\tilde{R}'_{k,k}|^2 = |\tilde{R}_{k-1,k-1}|^2 + |\tilde{R}_{k-1,k}|^2 + |\tilde{R}_{k,k}|^2. \tag{B.2}$$

For the updated $\tilde{R}'_{k-1,k}$, based on the operation in Lines 12-14 of Table 2.1, we have

$$|\tilde{R}'_{k-1,k}|^2 = \left| \frac{\tilde{R}^*_{k-1,k} \tilde{R}_{k-1,k-1}}{||\tilde{\boldsymbol{R}}_{k-1:k,k}||} \right|^2 = \frac{|\tilde{R}_{k-1,k}|^2 |\tilde{R}_{k-1,k-1}|^2}{|\tilde{R}_{k,k}|^2 + |\tilde{R}_{k-1,k}|^2} > \frac{|\tilde{R}_{k-1,k}|^2 |\tilde{R}_{k-1,k-1}|^2}{|\tilde{R}_{k-1,k-1}|^2} = |\tilde{R}_{k-1,k}|^2, \tag{B.3}$$

where the inequality comes from (B.1). Similarly, the updated $\tilde{R}'_{k,k}$ satisfies

$$|\tilde{R}'_{k,k}|^2 = \left| \frac{-\tilde{R}_{k-1,k-1} \tilde{R}_{k,k}}{||\tilde{\boldsymbol{R}}_{k-1:k,k}||} \right|^2 > |\tilde{R}_{k,k}|^2. \tag{B.4}$$

Since the summation in (B.2) keeps unchanged while both $|\tilde{R}'_{k-1,k}|^2$ and $|\tilde{R}'_{k,k}|^2$ increase, we obtain that $|\tilde{R}'_{k-1,k-1}|^2$ decreases and the decreased value from $|\tilde{R}'_{k-1,k-1}|^2$ is larger than the increased value from $|\tilde{R}'_{k,k}|^2$. This completes the proof.

APPENDIX C

PROOF OF PROPOSITION 4.1

. In order to derive the diversity order in the dual-LR-aided LDs with the proposed two greedy LLL algorithms, we introduce a parameter to quantify the orthogonality of the channel matrix $\boldsymbol{H}$, i.e.,

Definition C.1 (Orthogonality Deficiency [41]): For a given $N_r \times N_t$ channel matrix $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_{N_t}]$, the orthogonality deficiency (od) is defined as

$$od(\boldsymbol{H}) = 1 - \frac{\det(\boldsymbol{H}^{\mathcal{H}}\boldsymbol{H})}{\prod_{k=1}^{N_t} ||\boldsymbol{h}_k||^2}. \tag{C.1}$$

Note that $od(\boldsymbol{H}) = 0$ if $\boldsymbol{H}$ has orthogonal columns while $od(\boldsymbol{H}) = 1$ if $\boldsymbol{H}$ is singular. Therefore, $0 \leq od(\boldsymbol{H}) \leq 1, \forall \boldsymbol{H}$. In general, LR algorithms can reduce $od(\boldsymbol{H})$ to make the channel closer to orthogonal, such that better error performance can be obtained in the (dual-)LR-aided LDs.

For the reduced dual basis $\tilde{\boldsymbol{H}}^{\star} = \tilde{\boldsymbol{Q}}^{\star}\tilde{\boldsymbol{R}}^{\star}$ with the proposed two greedy LLL algorithms, based on the relaxed Lovász condition (4.2), we have

$$|\tilde{R}^{\star}_{k-2,k-2}|^2 \leq 2|\tilde{R}^{\star}_{k-1,k-1}|^2 \leq 2^2|\tilde{R}^{\star}_{k,k}|^2, \quad 3 \leq k \leq N_t, \tag{C.2}$$

which can be generalized as

$$|\tilde{R}^{\star}_{i,i}|^2 \leq 2^{k-i}|\tilde{R}^{\star}_{k,k}|^2, \quad 1 \leq i < k \leq N_t. \tag{C.3}$$

Then, together with the results from size reduction in (3.6), the column norm of $\tilde{\boldsymbol{H}}^{\star}$ satisfies

$$\begin{aligned} ||\tilde{\boldsymbol{h}}^{\star}_k||^2 &= \sum_{i=1}^{k} |\tilde{R}^{\star}_{i,k}|^2 \leq |\tilde{R}^{\star}_{k,k}|^2 + \sum_{i=1}^{k-1} \frac{1}{2}|\tilde{R}^{\star}_{i,i}|^2 \\ &\leq |\tilde{R}^{\star}_{k,k}|^2 + \sum_{i=1}^{k-1} 2^{k-i-1}|\tilde{R}^{\star}_{k,k}|^2 < 2^{k-1}|\tilde{R}^{\star}_{k,k}|^2. \end{aligned} \tag{C.4}$$

Therefore, the orthogonality deficiency of the reduced dual basis $\tilde{\boldsymbol{H}}^\star$ satisfies

$$\begin{aligned} od(\tilde{\boldsymbol{H}}^\star) &= 1 - \frac{\det\left((\tilde{\boldsymbol{H}}^\star)^{\mathcal{H}}\tilde{\boldsymbol{H}}^\star\right)}{\prod_{k=1}^{N_t}||\boldsymbol{h}_k^\star||^2} = 1 - \frac{\prod_{k=1}^{N_t}|\tilde{R}_{k,k}^\star|^2}{\prod_{k=1}^{N_t}||\boldsymbol{h}_k^\star||^2} \\ &< 1 - \frac{\prod_{k=1}^{N_t}|\tilde{R}_{k,k}^\star|^2}{\prod_{k=1}^{N_t}2^{k-1}|\tilde{R}_{k,k}^\star|^2} = 1 - 2^{-\frac{N_t(N_t-1)}{2}}. \end{aligned} \tag{C.5}$$

Once we get $od(\tilde{\boldsymbol{H}}^\star)$, from [63, Lemma 1], we obtain an upper bound for the column norm of the reduced prime basis $\tilde{\boldsymbol{H}}$ as

$$||\tilde{\boldsymbol{h}}_k|| \leq \frac{1}{\sqrt{1 - od(\tilde{\boldsymbol{H}}^\star)}||\tilde{\boldsymbol{h}}_i^\star||}, \quad k = N_t - i + 1. \tag{C.6}$$

Furthermore, by multiplying all column norms of $\tilde{\boldsymbol{H}}$, we have

$$\prod_{k=1}^{N_t}||\tilde{\boldsymbol{h}}_k||^2 \leq \frac{1}{\left(1 - od(\tilde{\boldsymbol{H}}^\star)\right)^{N_t}\prod_{k=1}^{N_t}||\tilde{\boldsymbol{h}}_k^\star||^2}. \tag{C.7}$$

Based on (C.5) and (C.7), and $\boldsymbol{H}^\star = (\boldsymbol{H}^\dagger)^{\mathcal{H}}\boldsymbol{J}$, we obtain an upper bound for the orthogonality deficiency of the reduced prime basis $\tilde{\boldsymbol{H}}$ as

$$\begin{aligned} od(\tilde{\boldsymbol{H}}) &= 1 - \frac{\det\left(\left(((\tilde{\boldsymbol{H}}^\star)^\dagger)^{\mathcal{H}}\boldsymbol{J}\right)^{\mathcal{H}}((\tilde{\boldsymbol{H}}^\star)^\dagger)^{\mathcal{H}}\boldsymbol{J}\right)}{\prod_{k=1}^{N_t}||\boldsymbol{h}_k^\star||^2} = 1 - \frac{\left(\det\left((\tilde{\boldsymbol{H}}^\star)^{\mathcal{H}}\tilde{\boldsymbol{H}}^\star\right)\right)^{-1}}{\prod_{k=1}^{N_t}||\boldsymbol{h}_k^\star||^2} \\ &\leq 1 - \frac{\left(1 - od(\tilde{\boldsymbol{H}}^\star)\right)^{N_t}\prod_{k=1}^{N_t}||\tilde{\boldsymbol{h}}_k^\star||^2}{\det\left((\tilde{\boldsymbol{H}}^\star)^{\mathcal{H}}\tilde{\boldsymbol{H}}^\star\right)} \\ &= 1 - \left(1 - od(\tilde{\boldsymbol{H}}^\star)\right)^{N_t-1} < 1 - 2^{-\frac{N_t(N_t-1)^2}{2}}. \end{aligned} \tag{C.8}$$

Based on (C.8), an equivalent inequality can be obtained as

$$\sqrt{1 - od(\tilde{\boldsymbol{H}})} > 2^{-\frac{N_t(N_t-1)^2}{4}}, \tag{C.9}$$

which indicates that $\sqrt{1 - od(\tilde{\boldsymbol{H}})}$ has a positive lower bound that is less than 1 for any integer $N_t > 1$. It has been proven that the symbol error probability of the LR-aided LDs for a given channel matrix $\boldsymbol{H}$ has the following relationship with respect to $\sqrt{1 - od(\tilde{\boldsymbol{H}})}$ [41]

$$P_{e|\boldsymbol{H}} \leq P\left(\frac{||\boldsymbol{w}||}{\sqrt{1 - od(\tilde{\boldsymbol{H}})}\min_{1\leq i\leq N_t}||\tilde{\boldsymbol{h}}_i||} \geq 1 \middle| \boldsymbol{H}\right). \tag{C.10}$$

Therefore, the average symbol error probability can be obtained by averaging (C.10) with respect to $\boldsymbol{H}$. Since we have obtained a lower bound for $\sqrt{1-od(\tilde{\boldsymbol{H}})}$ as shown (C.9), following [41, Proposition 1], we can derive the upper bound of the average symbol error probability as

$$P_e \leq C_{N_t N_r} \left(\frac{1}{\sigma_w^2}\right)^{-N_r}, \tag{C.11}$$

where $C_{N_t N_r}$ is a finite constant depending on N_t and N_r as

$$C_{N_t N_r} = \left(\sum_{n=1}^{\infty} \frac{1}{n^{N_r - N_t + \frac{1}{2}}}\right) \frac{\pi^{N_t}(2N_r - 1)!}{(N_r - 1)!(N_t - 1)!} 2^{\frac{N_r N_t (N_t-1)^2 + 2}{2}}. \tag{C.12}$$

Therefore, based on (C.11), the dual-LR-aided LDs with the proposed two greedy LLL algorithms achieve the full receive diversity order N_r.

APPENDIX D

PROOF OF PROPOSITION 4.2

.

For the reduced prime basis $\tilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{T} = \tilde{\boldsymbol{Q}}\tilde{\boldsymbol{R}}$ with the proposed two enhanced greedy LLL algorithms, based on the relaxed Lovász condition (4.2), we obtain the following inequality

$$|\tilde{R}_{k,k}|^2 \geq 2^{-1}|\tilde{R}_{k-1,k-1}|^2, \quad 2 \leq k \leq N_t. \tag{D.1}$$

Applying this inequality recursively, we further obtain

$$|\tilde{R}_{k,k}|^2 \geq 2^{1-N_t}|\tilde{R}_{1,1}|^2, \quad 1 \leq k \leq N_t, \tag{D.2}$$

which indicates

$$\min_k |\tilde{R}_{k,k}|^2 \geq 2^{1-N_t}|\tilde{R}_{1,1}|^2. \tag{D.3}$$

Since we have

$$|\tilde{R}_{1,1}|^2 = ||\tilde{\boldsymbol{h}}_1||^2 \geq \min_{\boldsymbol{d}\in\mathcal{D}} ||\boldsymbol{H}\boldsymbol{d}||^2, \tag{D.4}$$

where $\mathcal{D} = \{\boldsymbol{d} = \boldsymbol{s} - \boldsymbol{s}' | \boldsymbol{s}, \boldsymbol{s}' \in \mathcal{S}^{N_t}, \boldsymbol{s} \neq \boldsymbol{s}'\}$, combining (D.3) and (D.4), we obtain

$$\min_k |\tilde{R}_{k,k}|^2 \geq 2^{1-N_t} \min_{\boldsymbol{d}\in\mathcal{D}} ||\boldsymbol{H}\boldsymbol{d}||^2. \tag{D.5}$$

In high SNR region, the symbol error probability of the LR-aided SIC detection for a given $\boldsymbol{H}$ can be connected with $\min_k |\tilde{R}_{k,k}|^2$ as follows [13]

$$P_{e|\boldsymbol{H}} \leq 1 - \prod_{k=1}^{N_t} \left(1 - \exp\left(-\frac{|\tilde{R}_{k,k}|^2}{4\sigma_w^2}\right)\right) \approx \exp\left(-\frac{\min_k |\tilde{R}_{k,k}|^2}{4\sigma_w^2}\right). \tag{D.6}$$

With the lower bound for $\min_k |\tilde{R}_{k,k}|^2$ in (D.5), the error probability is further bounded as

$$P_{e|\boldsymbol{H}} \leq \exp\left(-\frac{2^{1-N_t} \min_{\boldsymbol{d}\in\mathcal{D}} ||\boldsymbol{H}\boldsymbol{d}||^2}{4\sigma_w^2}\right) = \exp\left(-\frac{2^{-N_t-1}||\boldsymbol{h}_d||^2}{\sigma_w^2}\right), \tag{D.7}$$

where $||\boldsymbol{h}_d||^2 = \min_{\boldsymbol{d}\in\mathcal{D}} ||\boldsymbol{H}\boldsymbol{d}||^2$, and $\boldsymbol{h}_d$ is a linear combination of $\boldsymbol{h}_k$'s, $k \in [1, N_t]$, with coefficients $\boldsymbol{d}$ drawn from Gaussian integer ring. If $\boldsymbol{H}$ has i.i.d. entries, the entries in $\boldsymbol{h}_d$ are also i.i.d.. Thus, by averaging (D.7) with respect to $\boldsymbol{H}$, we obtain the average symbol error probability bound as [48, Ch. 14]

$$P_e \leq C_d \left(\frac{1}{\sigma_w^2}\right)^{-G_d}, \tag{D.8}$$

where C_d is a finite constant. The parameter G_d in (D.8) is calculated as [41, 70]

$$G_d = \text{rank}(E[\boldsymbol{h}_d \boldsymbol{h}_d^{\mathcal{H}}]) = \text{rank}(C_{\boldsymbol{h}_d} \boldsymbol{I}_{N_r}) = N_r, \tag{D.9}$$

where $C_{\boldsymbol{h}_d}$ is a finite constant depending on N_r and the variance of the entries in $\boldsymbol{h}_d$. The results of (D.8) and (D.9) indicate that the LR-aided SIC detectors with the enhanced greedy LLL algorithm also achieves the full receive diversity order N_r.

REFERENCES

[1] Agrell, E., Eriksson, T., Vardy, A., and Zeger, K., "Closest point search in lattices," *IEEE Trans. Inf. Theory*, vol. 48, pp. 2201–2214, Aug. 2002.

[2] Ahmad, U., Li, M., Appeltans, R., Nguyen, H. D., Amin, A., Dejonghe, A., Van der Perre, L., Lauwereins, R., and Pollin, S., "Exploration of lattice reduction aided soft-output MIMO detection on a DLP/ILP baseband processor," *IEEE Trans. Signal Process.*, vol. 61, pp. 5878–5892, Dec. 2013.

[3] Andraka, R., "A survey of CORDIC algorithms for FPGA based computers," in *Proc. ACM/SIGDA 6th Int. Symp. on Field programmable gate arrays (FPGA)*, (Monterey, CA), pp. 191–200, Feb. 1998.

[4] Bai, L. and Choi, J., "Lattice reduction-based MIMO iterative receiver using randomized sampling," *IEEE Trans. Wireless Commun.*, vol. 12, pp. 2160–2170, May 2013.

[5] Barbero, L. G., Milliner, D. L., Ratnarajah, T., Barry, J. R., and Cowan, C., "Rapid prototyping of Clarkson's lattice reduction for MIMO detection," in *Proc. IEEE Int. Conf. Commun. (ICC)*, (Dresden, Germany), pp. 1–5, June 2009.

[6] Barbero, L. G., Ratnarajah, T., and Cowan, C., "A comparison of complex lattice reduction algorithms for MIMO detection," in *Proc. IEEE Int. Conf. Acoust., Speech and Signal Process. (ICASSP)*, (Las Vegas, NV), pp. 2705–2708, Mar. 2008.

[7] Bj, E., Larsson, E. G., Marzetta, T. L., and others, "Massive MIMO: Ten myths and one critical question," *IEEE Commun. Mag.*, vol. 54, pp. 114–123, Feb. 2016.

[8] Boccardi, F., Heath Jr, R. W., Lozano, A., Marzetta, T. L., and Popovski, P., "Five disruptive technology directions for 5G," *IEEE Commun. Mag.*, vol. 52, pp. 74–80, Feb. 2014.

[9] Bruderer, L., Studer, C., Wenk, M., Seethaler, D., and Burg, A., "VLSI implementation of a low-complexity LLL lattice reduction algorithm for MIMO detection," in *Proc. IEEE Int. Symp. on Circuits and Syst. (ISCAS)*, (Paris, France), pp. 3745–3748, June 2010.

[10] Burg, A., Seethaler, D., and Matz, G., "VLSI implementation of a lattice-reduction algorithm for multi-antenna broadcast precoding," in *IEEE Int. Symp. on Circuits and Syst. (ISCAS)*, (New Orleans, LA), pp. 673–676, May 2007.

[11] Cantin, M., Savaria, Y., and Lavoie, P., "A comparison of automatic word length optimization procedures," in *IEEE Int. Symp. on Circuits and Syst. (ISCAS)*, vol. 2, (Seoul, Korea), pp. 612–615, May 2002.

[12] Chen, C.-E., Cho, T.-W., and Chung, W.-H., "Blockwise-lattice-reduction-aided Tomlinson–Harashima precoder designs for MU-MIMO downlink communications with clusters of correlated users," *IEEE Trans. Veh. Technol.*, vol. 63, pp. 1146–1159, Mar. 2014.

[13] Choi, J. and Adachi, F., "User selection criteria for multiuser systems with optimal and suboptimal LR based detectors," *IEEE Trans. Signal Process.*, vol. 58, pp. 5463–5468, Oct. 2010.

[14] Cmar, R., Rijnders, L., Schaumont, P., Vernalde, S., and Bolsens, I., "A methodology and design environment for DSP ASIC fixed point refinement," in *Proc. Design, Automation and Test in Europe Conf. and Exhib.*, (Munich, Germany), pp. 271–276, Mar. 1999.

[15] Constantinides, G. and Woeginger, G., "The complexity of multiple wordlength assignment," *Applied Mathematics Letters*, vol. 15, no. 2, pp. 137–140, 2002.

[16] Di Renzo, M., Haas, H., Ghrayeb, A., Sugiura, S., and Hanzo, L., "Spatial modulation for generalized MIMO: Challenges, opportunities, and implementation," *Proc. IEEE*, vol. 102, pp. 56–103, Jan. 2014.

[17] Gan, Y. H., Ling, C., and Mow, W. H., "Complex lattice reduction algorithm for low-complexity full-diversity MIMO detection," *IEEE Trans. Signal Process.*, vol. 57, pp. 2701–2710, July 2009.

[18] Gestner, B., Zhang, W., Ma, X., and Anderson, D., "Lattice reduction for MIMO detection: From theoretical analysis to hardware realization," *IEEE Trans. Circuits Syst. I*, vol. 58, pp. 813–826, Apr. 2011.

[19] Gestner, B., Zhang, W., Ma, X., and Anderson, D. V., "VLSI implementation of a lattice reduction algorithm for low-complexity equalization," in *Proc. IEEE Int. Conf. Circuits Syst. Commun. (ICCSC)*, (Shanghai, China), pp. 643–647, May 2008.

[20] Heckler, C. and Thiele, L., "Complexity analysis of a parallel lattice basis reduction algorithm," *SIAM J. Comput.*, vol. 27, pp. 1295–1302, Oct. 1998.

[21] Howgrave-Graham, N., "Finding small roots of univariate modular equations revisited," in *Proc. the 6th IMA Int. conf. on Crypt. and Coding (IMACC)*, (Cirencester, UK), pp. 131–142, Dec. 1997.

[22] Jaldén, J. and Ottersten, B., "On the complexity of sphere decoding in digital communications," *IEEE Trans. Signal Process.*, vol. 53, pp. 1474–1484, Apr. 2005.

[23] Jaldén, J. and Elia, P., "DMT optimality of LR-aided linear decoders for a general class of channels, lattice designs, and system models," *IEEE Trans. Inf. Theory*, vol. 56, pp. 4765–4780, Oct. 2010.

[24] Jaldén, J., Seethaler, D., and Matz, G., "Worst-and average-case complexity of LLL lattice reduction in MIMO wireless systems," in *Proc. IEEE Int. Conf. Acoust., Speech and Signal Process.(ICASSP)*, (Las Vegas, NV), pp. 2685–2688, Mar. 2008.

[25] Jiang, H. and Du, S., "Complex Korkine-Zolotareff reduction algorithm for full-diversity MIMO detection," *IEEE Commun. Lett.*, vol. 17, pp. 381–384, Feb. 2013.

[26] Lagarias, J. C., Lenstra Jr, H. W., and Schnorr, C.-P., "Korkin-Zolotarev bases and successive minima of a lattice and its reciprocal lattice," *Combinatorica*, vol. 10, no. 4, pp. 333–348, 1990.

[27] Larsson, E. G., "MIMO detection methods: How they work," *IEEE Signal Process. Mag.*, vol. 26, pp. 91–95, May 2009.

[28] Larsson, E. G., Edfors, O., Tufvesson, F., and Marzetta, T. L., "Massive MIMO for next generation wireless systems," *IEEE Commun. Mag.*, vol. 52, pp. 186–195, Feb. 2014.

[29] Lenstra, A. K., Lenstra, H. W., and Lovász, L., "Factoring polynomials with rational coefficients," *Math. Annalen*, vol. 261, no. 4, pp. 515–534, 1982.

[30] Liao, C.-F. and Huang, Y.-H., "Power-saving 4×4 lattice-reduction processor for MIMO detection with redundancy checking," *IEEE Trans. Circuits Syst. II*, vol. 58, pp. 95–99, Feb. 2011.

[31] Ling, C., "On the proximity factors of lattice reduction-aided decoding," *IEEE Trans. Signal Process.*, vol. 59, pp. 2795–2808, June 2011.

[32] Ling, C., "Approximate lattice decoding: Primal versus dual basis reduction," in *Proc. IEEE Int. Symp. Info. Theory (ISIT)*, (Seattle,WA), pp. 1–5, July 2006.

[33] Ling, C., "Improved upper bounds for approximate lattice decoding with dual-basis reduction," in *Proc. IEEE Int. Conf. Commun. (ICC)*, (Beijing, China), pp. 1181–1184, May 2008.

[34] Ling, C. and Howgrave-Graham, N., "Effective LLL reduction for lattice decoding," in *Proc. IEEE Int. Symp. Info. Theory (ISIT)*, (Nice, France), pp. 196–200, June 2007.

[35] Ling, C., Mow, W. H., and Gan, L., "Dual-lattice ordering and partial lattice reduction for SIC-based MIMO detection," *IEEE J. Sel. Topics Signal Process.*, vol. 3, pp. 975–985, Dec. 2009.

[36] Ling, C., Mow, W. H., and Howgrave-Graham, N., "Reduced and fixed-complexity variants of the LLL algorithm for communications," *IEEE Trans. Commun.*, vol. 61, pp. 1040–1050, Mar. 2013.

[37] Lu, L., Li, G. Y., Swindlehurst, A. L., Ashikhmin, A., and Zhang, R., "An overview of massive MIMO: Benefits and challenges," *IEEE J. Sel. Topics Signal Process.*, vol. 8, pp. 742–758, Oct. 2014.

[38] Ma, X. and Zhang, W., "Performance analysis for MIMO systems with lattice-reduction aided linear equalization," *IEEE Trans. Commun.*, vol. 56, pp. 309–318, Feb. 2008.

[39] Ma, X. and Wen, Q., "Multi-input multi-output (MIMO) detection systems," in *US Patent Application No. 15/056986, Publication No. US20160254883 A1*, Sept. 2016.

[40] Ma, X. and Zhang, W., "Fundamental limits of linear equalizers: Diversity, capacity, and complexity," *IEEE Trans. Inf. Theory*, vol. 54, pp. 3442–3456, Aug. 2008.

[41] Ma, X. and Zhang, W., "Performance analysis for MIMO systems with lattice-reduction aided linear equalization," *IEEE Trans. Commun.*, vol. 56, pp. 309–318, Feb. 2008.

[42] Ma, X. and Zhou, Q., "Massive MIMO and its detection," in *MIMO Processing for 4G and Beyond: Fundamentals and Evolution* (da Silva, M. M. and Monteiro, F. A., eds.), ch. 10, pp. 449–471, Boca Raton, FL: CRC Press, 2014.

[43] Mietzner, J., Schober, R., Lampe, L., Gerstacker, W. H., and Hoeher, P. A., "Multiple-antenna techniques for wireless communications - a comprehensive literature survey," *IEEE Commu. Surveys & Tutorials*, vol. 11, pp. 87–105, Second Quart. 2009.

[44] Mow, W. H., "Universal lattice decoding: A review and some recent results," in *Proc. IEEE Int. Conf. Commun. (ICC)*, (Paris, France), pp. 2842–2846, June 2004.

[45] Murugan, A., El Gamal, H., Damen, M. O., and Caire, G., "A unified framework for tree search decoding: Rediscovering the sequential decoder," *IEEE Trans. Inf. Theory*, vol. 52, pp. 933–953, Mar. 2006.

[46] Parashar, K., Rocher, R., Menard, D., and Sentieys, O., "A hierarchical methodology for word-length optimization of signal processing systems," in *Int. Conf. on VLSI Design (VLSID)*, (Bangalore, India), pp. 318–323, Jan. 2010.

[47] Paulraj, A., Gore, D., Nabar, R., and Bolcskei, H., "An overview of MIMO communications-a key to gigabit wireless," *Proc. IEEE*, vol. 92, pp. 198–218, Feb. 2004.

[48] Proakis, J. G., *Digital Communications*. New York: McGraw-Hill, Inc., 4th ed. ed., 2001.

[49] Qi, X.-F. and Holt, K., "A lattice-reduction-aided soft demapper for high-rate coded MIMO-OFDM systems," *IEEE Signal Process. Lett.*, vol. 14, pp. 305–308, May 2007.

[50] Rusek, F., Persson, D., Lau, B., Larsson, E., Marzetta, T., Edfors, O., and Tufvesson, F., "Scaling up MIMO: Opportunities and challenges with very large arrays," *IEEE Signal Process. Mag.*, vol. 30, pp. 40–60, Jan. 2013.

[51] Seethaler, D., Matz, G., and Hlawatsch, F., "Low-complexity MIMO data detection using Seysen's lattice reduction algorithm," in *Proc. IEEE Int. Conf. Acoust., Speech and Signal Processing (ICASSP)*, (Honolulu, HI), pp. 53–56, Apr. 2007.

[52] Senning, C., Bruderer, L., Hunziker, J., and Burg, A., "A lattice reduction-aided MIMO channel equalizer in 90 nm CMOS achieving 720 Mb/s," *IEEE Trans. Circuits Syst. I*, vol. 61, pp. 1860–1871, June 2014.

[53] Seysen, M., "Simultaneous reduction of a lattice basis and its reciprocal basis," *Combinatorica*, vol. 13, pp. 363–376, Sept. 1993.

[54] Shabany, M. and Glenn Gulak, P., "The application of lattice-reduction to the K-best algorithm for near-optimal MIMO detection," in *Proc. IEEE Int. Symp. on Circuits and Syst. (ISCAS)*, (Seattle, WA), pp. 316–319, May 2008.

[55] Shabany, M., Youssef, A., and Gulak, G., "High-throughput 0.13-CMOS lattice reduction core supporting 880 Mb/s detection," *IEEE Trans. VLSI Syst.*, vol. 21, pp. 848–861, May 2013.

[56] Shahabuddin, S., Janhunen, J., Ghazi, A., Khan, Z., and Juntti, M., "A customized lattice reduction multiprocessor for MIMO detection," in *Proc. IEEE Int. Symp. on Circuits and Syst. (ISCAS)*, (Lisbon, Portugal), pp. 2976–2979, May 2015.

[57] Sheikh, F., Balatsoukas-Stimming, A., and Chen, C.-H., "High-throughput lattice reduction for large-scale MIMO systems based on Seysen's algorithm," in *Proc. IEEE Int. Conf. Commun. (ICC)*, (Kuala Lumpur, Malaysia), pp. 1–6, May 2016.

[58] Silvola, P., Hooli, K., and Juntti, M., "Suboptimal soft-output MAP detector with lattice reduction," *IEEE Signal Process. Lett.*, vol. 13, p. 321, June 2006.

[59] Singhal, K. A., Datta, T., and Chockalingam, A., "Lattice reduction aided detection in large-MIMO systems," in *Proc. IEEE 14th Workshop on Signal Process. Adv. in Wireless Commun. (SPAWC)*, (Darmstadt, Germany), pp. 594–598, June 2013.

[60] Soler-Garrido, J., Vetter, H., Sandell, M., Milford, D., and Lillie, A., "Implementation of a reduced-lattice MIMO detector for OFDM systems," in *Proc. Conf. Design, Automation and Test in Europe (DATE)*, (Nice, France), pp. 1626–1631, Apr. 2009.

[61] Sung, W. and Kum, K.-I., "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE Trans. Signal Process.*, vol. 43, no. 12, pp. 3087–3090, 1995.

[62] Taherzadeh, M. and Khandani, A., "On the limitations of the naive lattice decoding," *IEEE Trans. Inf. Theory*, vol. 56, pp. 4820–4826, Oct. 2010.

[63] Taherzadeh, M., Mobasher, A., and Khandani, A., "LLL reduction achieves the receive diversity in MIMO decoding," *IEEE Trans. Inf. Theory*, vol. 53, pp. 4801–4805, Dec. 2007.

[64] Tarokh, V., Jafarkhani, H., and Calderbank, A. R., "Space-time block coding for wireless communications: Performance results," *IEEE J. Sel. Areas Commun.*, vol. 17, pp. 451–460, Mar. 1999.

[65] Tse, D. and Viswanath, P., *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[66] Vetter, H., Ponnampalam, V., Sandell, M., and Hoeher, P. A., "Fixed complexity LLL algorithm," *IEEE Trans. Signal Process.*, vol. 57, pp. 1634–1637, Apr. 2009.

[67] Villard, G., "Parallel lattice basis reduction," in *Proc. ACM Int. Symp. on Symbolic and Algebraic Computation (ISSAC)*, (Berkeley, CA), pp. 269–277, July 1992.

[68] Wang, C., Cheng, P., Chen, Z., Zhang, J. A., Xiao, Y., and Gui, L., "Near-ML low-complexity detection for generalized spatial modulation," *IEEE Commun. Lett.*, vol. 20, pp. 618–621, Mar. 2016.

[69] Wang, N.-C., Biglieri, E., and Yao, K., "Systolic arrays for lattice-reduction-aided MIMO detection," *J. of Commun. and Networks*, vol. 13, pp. 481–493, Oct. 2011.

[70] Wang, Z. and Giannakis, G. B., "Complex-field coding for OFDM over fading wireless channels," *IEEE Trans. Inf. Theory*, vol. 49, no. 3, pp. 707–720, 2003.

[71] Wen, Q. and Ma, X., "An efficient greedy LLL algorithm for MIMO detection," in *Proc. Military Commu. Conf. (MILCOM)*, (Baltimore, MD), pp. 550–555, Oct. 2014.

[72] Wen, Q. and Ma, X., "An enhanced fixed-complexity LLL algorithm for MIMO detection," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, (Austin, TX), pp. 3231–3236, Dec. 2014.

[73] Wen, Q. and Ma, X., "High efficient wireless detectors with incremental fixed-complexity LLL algorithms," in *submitted disclosure*, Nov. 2014.

[74] Wen, Q. and Ma, X., "Efficient greedy LLL algorithms for lattice decoding," *IEEE Trans. Wireless Commun.*, vol. 15, pp. 3560–3572, May 2016.

[75] Wen, Q. and Ma, X., "Incremental fixed-complexity LLL algorithms," in *preparation for IEEE Trans. Signal Process.*, Dec. 2016.

[76] Wen, Q. and Ma, X., "VLSI implementation of incremental fixed-complexity LLL lattice reduction for MIMO detection," in *Proc. IEEE Int. Symp. on Circuits and Syst. (ISCAS)*, (Montreal, Canada), pp. 1898–1901, May 2016.

[77] Wen, Q. and Ma, X., "High throughput implementation of incremental fixed-complexity LLL algorithm," in *preparation for IEEE Trans. Circuits Syst. I*, Feb. 2017.

[78] Wen, Q. and Ma, X., "Low-complexity iterative implementation of incremental fixed-complexity LLL algorithm for MIMO detection," in *preparation for IEEE Trans. Circuits Syst. II*, Feb. 2017.

[79] Wen, Q., Zhou, Q., Zhao, C., and Ma, X., "Fixed-point realization of lattice-reduction aided MIMO receivers with complex K-best algorithm," in *Proc. IEEE Int. Conf. Acoust., Speech and Signal Process. (ICASSP)*, (Vancouver, Canada), pp. 5031–5035, May 2013.

[80] Windpassinger, C. and Fischer, R., "Low-complexity near-maximum-likelihood detection and precoding for MIMO systems using lattice reduction," in *Proc. IEEE Info. Theory Workshop (ITW)*, (Paris, France), pp. 345–348, Mar. 2003.

[81] Windpassinger, C., Fischer, R. F., and Huber, J. B., "Lattice-reduction-aided broadcast precoding," *IEEE Trans. Commun.*, vol. 52, pp. 2057–2060, Dec. 2004.

[82] Wu, D., Eilert, J., and Liu, D., "A programmable lattice-reduction aided detector for MIMO-OFDMA," in *Proc. IEEE Int. Conf. Circuits and Syst. for Commun. (ICCSC)*, (Shanghai, China), pp. 293–297, May 2008.

[83] Wübben, D., Böhnke, R., Kühn, V., and Kammeyer, K. D., "MMSE extension of V-BLAST based on sorted QR decomposition," in *Proc. IEEE 58th Veh. Technology Conf. (VTC2003-Fall)*, (Orlando, FL), pp. 508–512, Oct. 2003.

[84] Wübben, D., Böhnke, R., Kühn, V., and Kammeyer, K. D., "Near-maximum-likelihood detection of MIMO systems using MMSE-based lattice reduction," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 2, (Paris, France), pp. 798–802, June 2004.

[85] Wübben, D. and Seethaler, D., "On the performance of lattice reduction schemes for MIMO data detection," in *Proc. Asilomar Conf. Signals, Syst. and Comput.*, (Pacific Grove, CA), pp. 1534–1538, Nov. 2007.

[86] Wübben, D., Seethaler, D., Jaldén, J., and Matz, G., "Lattice reduction," *IEEE Signal Process. Mag.*, vol. 28, pp. 70–91, May 2011.

[87] Xilinx, Inc., "Virtex-5 FPGA User Guide, UG190 (v5.4)," *https://www.xilinx.com/support/documentation/user_guides/ug190.pdf*, Mar. 2012.

[88] Yang, H. J., Chun, J., Choi, Y., Kim, S., and Paulraj, A., "Codebook-based lattice-reduction-aided precoding for limited-feedback coded MIMO systems," *IEEE Trans. Commun.*, vol. 60, no. 2, pp. 510–524, 2012.

[89] Yang, P., Xiao, Y., Guan, Y. L., Hari, K. V. S., Chockalingam, A., Sugiura, S., Haas, H., Renzo, M. D., Masouros, C., Liu, Z., Xiao, L., Li, S., and Hanzo, L., "Single-carrier SM-MIMO: A promising design for broadband large-scale antenna systems," *IEEE Commun. Surveys Tuts.*, vol. 18, pp. 1687–1716, Third Quart. 2016.

[90] Yang, S. and Hanzo, L., "Fifty years of MIMO detection: The road to large-scale MIMOs," *IEEE Commun. Surveys Tuts.*, vol. 17, pp. 1941–1988, Fourth Quart. 2015.

[91] Yao, H. and Wornell, G. W., "Lattice-reduction-aided detectors for MIMO communication systems," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, (Taipei, Taiwan), pp. 424–428, Nov. 2002.

[92] Zhang, W., Arnold, F., and Ma, X., "An analysis of Seysen's lattice reduction algorithm," *Signal Processing*, vol. 88, pp. 2573–2577, Oct. 2008.

[93] Zhang, W. and Ma, X., "Lattice Reduction Aided Equalization for Wireless Applications," in *The Digital Signal Processing Handbook* (Madisetti, V., ed.), ch. 30, pp. 30–1–30–16, Boca Raton, FL: CRC Press, second ed., 2009.

[94] Zhang, W. and Ma, X., "Low-complexity soft-output decoding with lattice-reduction-aided detectors," *IEEE Trans. Commun.*, vol. 58, pp. 2621–2629, Sept. 2010.

[95] Zhang, W., Ma, X., Gestner, B., and Anderson, D. V., "Designing low-complexity equalizers for wireless systems," *IEEE Commun. Mag.*, vol. 47, pp. 56–62, Jan. 2009.

[96] Zhang, W., Ma, X., and Swami, A., "Designing low-complexity detectors based on Seysen's algorithm," *IEEE Trans. Wireless Commun.*, vol. 9, pp. 3301–3311, Oct. 2010.

[97] Zhang, W., Qiao, S., and Wei, Y., "A diagonal lattice reduction algorithm for MIMO detection," *IEEE Signal Process. Lett.*, vol. 19, pp. 311–314, May 2012.

[98] Zhang, W., Qiao, S., and Wei, Y., "HKZ and Minkowski reduction algorithms for lattice-reduction-aided MIMO detection," *IEEE Trans. Signal Process.*, vol. 60, pp. 5963–5976, Nov. 2012.

[99] Zhao, K., Li, Y., Jiang, H., and Du, S., "A low complexity fast lattice reduction algorithm for MIMO detection," in *Proc. IEEE Int. Symp. Personal, Indoor, Mobile Radio Commun. (PIMRC)*, (Sydney, Australia), pp. 1612–1616, Sept. 2012.

[100] Zheng, K., Zhao, L., Mei, J., Shao, B., Xiang, W., and Hanzo, L., "Survey of large-scale MIMO systems," *IEEE Commun. Surveys Tuts.*, vol. 17, pp. 1738–1760, Third Quart. 2015.

[101] Zhou, Q. and Ma, X., "An improved LR-aided K-best algorithm for MIMO detection," in *Proc. IEEE Int. Conf. on Wireless Commun. and Signal Process. (WCSP)*, (Huangshan, China), pp. 1–5, Oct. 2012.

[102] Zhou, Q. and Ma, X., "Element-based lattice reduction algorithms for large MIMO detection," *IEEE J. Sel. Areas Commun.*, vol. 31, pp. 274–286, Feb. 2013.

[103] Zhou, Q. and Ma, X., "Improved element-based lattice reduction algorithms for wireless communications," *IEEE Trans. Wireless Commun.*, vol. 12, pp. 4414–4421, Sept. 2013.

[104] Zhou, Q. and Ma, X., "Hardware realizable lattice-reduction-aided detectors for large-scale MIMO systems," in *Proc. 22nd European Signal Process. Conf. (EUSIPCO)*, (Lisbon, Portugal), pp. 91–95, Sept. 2014.

[105] Zu, K. and de Lamare, R. C., "Low-complexity lattice reduction-aided regularized block diagonalization for MU-MIMO systems," *IEEE Commun. Lett.*, vol. 16, pp. 925–928, June 2012.

VITA

Qingsong Wen received the B.S. and M.S. degrees in Communication and Information Engineering from University of Electronic Science and Technology of China, Chengdu, China, in 2006 and 2009, respectively. He is currently working towards the Ph.D. degree in the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta. From 2009 to 2011, he was an Architect and DSP Engineer with Marvell, Inc. at Shanghai, China. In 2013 Summer, he was a System Engineer Intern with Qualcomm, Inc. at San Diego, CA. In 2016 Summer, he was a Data Science Intern with Huawei R&D USA at Santa Clara, CA. His research interests include communications and computing systems, signal processing, and VLSI/FPGA architecture design and implementation.